Universitat de Lleida
Escola Politècnica Superior

Treball Final de Màster

MÀSTER EN ENGINYERIA INFORMÀTICA

Escola Politècnica Superior

Universitat de Lleida

Mòdul d'Optimització per a Recursos del Transport

Adrià Vall-llaura Salas

Tutors: Antonio Llubes, Josep Lluís Lérida

Data: Juny 2017

Pròleg

Aquest projecte s'ha desenvolupat per donar solució a un problema de l'ordre del dia d'una empresa de transports.

Es basa en el disseny i implementació d'un model matemàtic que ha de permetre optimitzar i automatitzar el sistema de planificació de viatges de l'empresa.

Per tal de poder implementar l'algoritme s'han hagut de crear diversos mòduls que extreuen les dades del sistema ERP, les tracten, les envien a un servei web (REST) i aquest retorna un emparellament òptim entre els vehicles de l'empresa i les ordres dels clients.

La primera fase del projecte, la teòrica, ha estat llarga en comparació amb les altres. En aquesta fase s'ha estudiat l'estat de l'art en la matèria i s'han repassat molts dels models més importants relacionats amb el transport per comprendre'n les seves particularitats. Amb els conceptes ben estudiats, s'ha procedit a desenvolupar un nou model matemàtic adaptat a les necessitats de la lògica de negoci de l'empresa de transports objecte d'aquest treball.

Posteriorment s'ha passat a la fase d'implementació dels mòduls. En aquesta fase m'he trobat amb diferents limitacions tecnològiques degudes a l'antiguitat de l'ERP i a l'ús del sistema operatiu Windows. També han sorgit diferents problemes de rendiment que m'han fet redissenyar l'extracció de dades de l'ERP, el càlcul de distàncies i el mòdul d'optimització.

La millora més significativa en disminució de temps s'ha donat en el càlcul de distàncies, passant de 8 o 9 hores de càlcul a menys de 30 segons. Finalment, la disminució més notable en l'ús de recursos s'ha donat en el mòdul d'optimització, on en problemes grans, s'ha passat d'ocupar entre 7 i 8 Gb de memòria RAM a ocupar 1.5 Gb.

En la següent fase s'ha validat tot el sistema per comprovar els resultats. Amb l'ajuda del departament de planificació s'han anat revisant els resultats de diferents situacions reals. S'ha arribat a la conclusió que falten tres tipus de restriccions per tal que les optimitzacions retornades siguin útils pel dia a dia de l'empresa. La més important és la restricció de disponibilitat de temps de conducció dels xofers, on el model ha fallat en el 85% dels casos.

Per solucionar els problemes trobats a la validació he fet un estudi de futures millores i implementacions que s'han de fer per obtenir resultats satisfactoris, de forma més ràpida i consumint menys recursos informàtics.

Agraïments

En primer lloc vull agrair als meus tutors del Treball de Fi de Màster, en Josep Lluís i en Toni, tot el recolzament rebut durant el projecte. Des de l'inici m'han anat aclarint les indecisions sobre idees que han anat sorgint durant el projecte i les hem anat reconduint fins a arribar al resultat final.

També vull agrair el temps que ha dedicat al departament de planificació de l'empresa per analitzar els resultats del sistema d'optimització i donar-me les seves opinions i consells al respecte.

Finalment no em puc deixar d'anomenar l'empresa per la qual he desenvolupat aquest projecte, A.T. TROTA S.A. Agraeixo la confiança dipositada en mi per dur a terme un projecte tant important com aquest i del qual pot dependre el futur de la planificació dels viatges de l'empresa.

ÍNDEX

ÍNDEX DE FIGURES

ÍNDEX DE TAULES

1 Introducció

El sector del transport porta bastants anys creixent substancialment i adaptant-se a les noves necessitats. Es dóna la circumstància que aquestes han anat canviant bastant en els darrers 15 anys.

Des de sempre, en el món del transport, es sentia a parlar de viatges de càrrega completa. No va ser fins fa uns 15 anys que es va començar a parlar d'agrupar càrregues petites de diferents clients per omplir els remolcs i aprofitar millor els viatges (consolidacions i desconsolidacions de càrrega d'agrupatge). Avui en dia ja són poques les empreses que no tenen en compte aquesta forma de treballar, d'aquesta manera es pot aprofitar molt més la capacitat màxima dels remolcs als viatges tant nacionals com internacionals.

Aquestes noves formes de treballar sorgeixen per tal de maximitzar el benefici de l'empresa però són bastant més complicades de coordinar i de mantenir una traçabilitat. Això ha fet obrir els ulls a les empreses del sector i han vist que avui en dia és necessari invertir en I+D per a desenvolupar, implementar i millorar algoritmes d'optimització de rutes i d'assignació de recursos que tinguin en compte aquestes característiques.

En els pròxims punts d'aquest document s'explicarà com es troba actualment l'estat de l'art en la matèria, s'anomenaran els principals models que resolen els problemes del sector, s'explicarà que és l'optimització (o programació) matemàtica i les branques que té i es detallarà la branca a la qual pertany el problema descrit en aquest projecte.

1.1 Programació matemàtica

La programació matemàtica, també anomenada optimització o optimització matemàtica, es pot entendre com la selecció del millor element dins d'un conjunt d'elements respectant alguns criteris i/o restriccions.

Quan parlem d'optimitzar, ens estem referint a que volem **maximitzar** o **minimitzar** el valor d'una funció real. Aquesta funció, junt amb totes les restriccions necessàries, formaran el model matemàtic que s'haurà modelat a partir d'un problema real.

Existeix una gran varietat d'especialitzacions dins de la programació matemàtica. Alguns dels diferents camps són:

- Programació convexa: Estudia el cas on la funció a optimitzar es convexa o còncava i les restriccions son convexes. Dins d'aquesta especialitat hi ha diferents ramificacions, la més coneguda es la programació lineal on la funció i les restriccions estan representades per funcions lineals.
- Programació entera: En aquest cas la funció també és lineal però alguna part o totes les restriccions són enteres. En general és molt més difícil de solucionar que la programació lineal.
- Programació quadràtica: La funció objectiu té termes quadràtics, el conjunt factible s'ha d'especificar amb igualtats i desigualtats lineals.
- Programació fraccional: Estudia l'optimització de proporcions de dos funcions no lineals.
- Programació no lineal: Estudia el cas on la funció objectiu, les restriccions o les dos coses contenen parts no lineals.

- Programació estocàstica: En aquest cas algunes de les restriccions o paràmetres poden ser aleatoris.
- Programació robusta: Intenta capturar la incertesa de les dades subjacents al problema d'optimització. Intenta trobar solucions vàlides sota totes les possibles variants de les incerteses.
- Programació combinatòria: Estudia el cas on el conjunt de solucions factibles és discret o pot ser reduït a un.

Els models relacionats amb el món del transport (que s'explicaran en el següent apartat) pertanyen als grups de **programació combinatòria** i **programació entera**. La naturalesa d'aquests problemes permet resoldre'ls utilitzant el mètode Símplex que s'explicarà més endavant.

1.2 Models relacionats amb el transport

Com ens podem imaginar, ja fa molt temps que s'investiga com solucionar el problema de satisfer una sèrie de comandes per diferents punts geogràfics recorrent la distància més curta possible i utilitzant els mínims recursos per fer-ho.

El primer problema d'aquest estil formulat matemàticament va ser el TSP (Travelling Salesman Problem) al segle XIX. Des de llavors, i adaptant-se a les noves formes de transport, han anat sorgint variacions d'aquest problema. Això ha creat un gran nombre de models matemàtics, la majoria dels quals formen part de la classe de complexitat computacional NP-Hard.

Tot seguit una enumeració dels principals models d'aquest sector que existeixen avui en dia:

- TSP: Travelling Salesman Problem. Donada una llista de ciutats i les distàncies entre cada parell d'aquestes, s'ha de trobar la ruta mes curta possible per a que el viatjant visiti cada ciutat exactament un cop i torni a la ciutat d'origen.
- M-TSP: Multiple TSP. Es una generalització del model anterior on pot existir més d'un viatjant.
- VRP: Vehicle Routing Problem. Es tracta de trobar les rutes d'una flota de transports per donar servei a uns clients. El més habitual és intentar minimitzar els costos de l'operació, temps total de transport, distància recorreguda, temps d'espera, utilització de vehicles, etc. En aquesta modelització general, es té en compte que els vehicles surten i han de tornar obligatòriament a un magatzem.
- OVRP: Open VRP. En aquesta modalitat, els vehicles no tenen que retornar a cap magatzem un cop finalitzat el servei a un client.
- CVRP: Capacited VRP. És com el VRP però s'afegeixen restriccions que contemplen la capacitat dels vehicles. Aquesta capacitat ha de ser uniforme per a cada vehicle.
- HVRP: Heterogeneous VRP. És com l'anterior però amb vehicles de diferents capacitats.
- VRPB: Backhauls VRP. Es una variant del VRP tenint en compte que a cada ruta, totes les entregues s'han de fer abans de les recollides.
- DVRP: Dynamic VRP. Una variació del model VRP on al començament del viatge encara no es saben els detalls de totes les recollides/entregues i que s'ha d'anar re-calculant periòdicament per adaptar-se a les noves demandes.
- PVRP: Periodic VRP. És una generalització del model VRP on s'estén el període de planificació a M dies.

- SVRP: Stochastic VRP. Es basa en el model VRP però pot ser que les variables com els clients, demandes o les finestres de temps siguin aleatòries. S'acostumen a resoldre en varies etapes.
- VRPPD: Pickup and Delivery VRP. És un VRP en el qual es contempla la possibilitat que els clients tornin alguns productes. En aquesta modalitat es necessita tenir en compte que els productes que els clients tornen al vehicle han de cabre-hi.
- VRPTW: VRP with Time Windows. És una extensió del model VRP amb la restricció addicional de que per a cada client s'associa una finestra de temps on aquest ha de ser atès.
- MDVRPTW: Multiple Depot VRPTW. És una generalització del model VRPTW on l'empresa disposa de més d'un magatzem des d'on començar a repartir.
- PDP: Pickup and Delivery Problem. En aquest model els vehicles han de transportar cargaments des d'un origen a un destí sense transbordaments en localitzacions intermitges.
- PDPTW: PDP with Time Windows. És una extensió del model PDP amb la restricció addicional de que per a cada client s'associa una finestra de temps on el client ha de ser atès.
- MDPDPTW: Multiple Depot PDPTW. És una generalització del model PDPTW on l'empresa disposa de més d'un magatzem des d'on començar a repartir.
- MDVRPPDTW: MDVRP Pickup and Delivery with Time Windows. És una generalització del model VRPPDTW on l'empresa disposa de més d'un magatzem des d'on començar a repartir.

Com es pot comprovar en aquests models, n'hi ha tres de bàsics. La resta de models són els mateixos però incorporen més restriccions o generalitzacions per adaptar-se millor a la realitat del sector.

Si hagués d'escollir un model concret per descriure el problema que s'intenta solucionar en aquest projecte, no em serviria cap dels mencionats anteriorment. Els models matemàtics "estàndard" estan modelats de forma teòrica però a la pràctica cada empresa adopta un model de negoci que pot fer variar substancialment els models, com en el nostre cas.

Per acabar definirem una nomenclatura que podria encaixar al model de negoci de la meva empresa:

- DCPOMDVRPTW: Dynamic Capacited Periodic Open Multiple Depot Vehicle Routing Problem with Time Windows.

Amb aquest model tampoc descriuria completament el nostre problema, sinó que encara faltarien afegir restriccions de canvis de remolc, disponibilitat de conducció de xofers, cabotatges i restriccions del taller entre d'altres.

Intentar **solucionar aquest problema de cop és inviable** tenint en compte que **es necessita una solució en un temps raonable**, així doncs, com es descriurà més endavant, s'ha simplificat el model i s'ha optat per centrar el projecte en un model del tipus MDPDPTW(Multiple Depot Pickup and Delivery Problem with Time windows).

1.3 Estat de l'art

Fins fa pocs anys la tendència era trobar nous models d'optimització que s'adaptessin a les necessitats canviants del sector. Actualment aquesta tendència està evolucionant ja que s'han trobat pràcticament tots els models teòrics estàndard. Ara mateix el problema està en

implementar-los de forma que siguin útils en el món empresarial. Això vol dir que s'han de poder resoldre en un temps raonable (de minuts o poques hores, segons el problema) i que han de consumir la mínima quantitat de recursos.

Conseqüentment existeix un nombre elevat d'articles acadèmics i de tesis on s'estudien diferents tipus d'heurístiques i metaheurístiques per descompondre grans problemes del transport en problemes més assequibles. També hi ha estudis on s'implementen nous models matemàtics i on es milloren models ja existents.

En la secció 1.2 ja s'han comentat els models més representatius que s'utilitzen actualment, per tant en aquest apartat em centraré en tres punts bàsics: Algoritme Símplex, Heurístiques i metaheurístiques i algoritmes altament paral·lelitzables.

1.3.1 Algoritme Símplex

Aquest algoritme és el que s'acostuma a fer servir per resoldre problemes d'optimització d'aquest tipus. És un procediment iteratiu que permet millorar la solució de la funció objectiu a cada pas. El procés acaba quan ja no és possible millorar la solució i per tant s'ha arribat al punt òptim.

Partint del valor de la funció objectiu en un punt qualsevol, el procediment consisteix en buscar un altre punt que millori el valor anterior. Aquests punts es representen com a vèrtex d'un polígon que constitueix la regió determinada per les restriccions a les que es troba la regió factible del problema. La recerca es realitza mitjançant desplaçaments per les arestes del polígon, des del vèrtex actual fins a un d'adjacent que millori el valor de la funció objectiu. Sempre que existeixi una regió factible, com el nombre de vèrtex és finit, serà possible trobar una solució.

Per fer servir aquest mètode és necessari estandarditzar les restriccions i les inequacions del problema per a que totes siguin del tipus "menor o igual (≤)" i els seus coeficients independents siguin majors o iguals a 0.

Existeixen dues variants bàsiques d'aquest mètode:

- Primal Símplex: Es fa servir quan es comença amb un problema que té una solució factible i bàsica però no és òptima. Aquest mètode busca la optimalitat i la solució final és bàsica, factible i òptima.
- Dual Símplex: Es fa servir quan es comença amb un problema que té una solució infactible, bàsica i òptima. Aquest mètode busca la factibilitat i la solució final és bàsica, factible i òptima.

Per comprendre millor les anteriors definicions, també cal definir els següents conceptes:

- Solució factible: És una solució per a la qual es satisfan totes les restriccions.
- Solució òptima: És una solució factible que té el valor més favorable de la funció objectiu.
- Solució bàsica: Per a un sistema de m equacions i n variables en el que $n>m$, si existeix una solució, pot trobar-se igualant $n-m$ variables a zero i resolent el conjunt resultant d'm equacions amb m variables. Les variables que s'igualen a zero es denominen variables no bàsiques; les variables que es fan servir per resoldre les equacions es denominen variables bàsiques. Una solució bàsica és un punt extrem.
- Solució bàsica factible: És una solució bàsica que també satisfà les condicions de no negativitat i per tant totes les variables bàsiques són no negatives.

Normalment els models relacionats amb el món del transport es resolen usant el mètode dual símplex degut que tenen un nombre de restriccions molt més elevat en comparació amb el nombre de variables de la funció objectiu, cosa que facilita la cerca de l'òptim.

Els apartats següents no s'han implementat en aquest projecte però és important comentar-los per saber cap a on apunten les investigacions avui en dia.

1.3.2 Heurístiques i metaheurístiques

L'aplicació d'heurístiques o metaheurístiques en problemes d'optimització es fa servir per descompondre un problema gran i complex en varis problemes més senzills i ràpids de resoldre computacionalment.

Contínuament s'implementen millores a aquests tipus d'algorismes i s'estudien noves heurístiques/metaheurístiques per adaptar-se als models més actuals del sector del transport.

1.3.3 Algoritmes altament paral·lelitzables

A mesura que van evolucionant els models i sorgeixen noves formes de treballar en el sector del transport, s'han de solucionar problemes més complexos i fer-ho de forma més ràpida per tal que les solucions siguin efectives.

Per agilitzar aquests processos s'estan estudiant solucions com per exemple paral·lelitzar l'algoritme símplex implementant-lo amb la tecnologia CUDA. D'aquesta forma s'aprofitaria el potencial dels clústers de unitats de processament gràfic (conjunt de màquines connectades entre sí per resoldre de forma col·laborativa problemes grans). El problema és que l'algoritme Símplex és molt complicat d'implementar en aquests entorns sense generar colls d'ampolla amb l'intercanvi de missatges entre el processador i les unitats gràfiques.

D'altra banda, la creació de noves metaheurístiques que descomponen la solució del problema originals en problemes més petits, afavoreixen l'aparició de tècniques que tracten d'explotar el paral·lelisme de la descomposició de les parts per optimitzar-ne tant el temps de resolució com augmentar l'escalabilitat dels problemes a resoldre.

1.4 Motivació i objectius

Aquest projecte s'ha plantejat i s'està duent a terme per donar solució a un problema real en l'empresa on treballo actualment (A.T Trota S.A).

Avui en dia disposem d'un departament de planificació que s'encarrega d'emparellar manualment els diferents recursos de transport de l'empresa. Aquesta planificació busca aprofitar al màxim els viatges, això vol dir que intenta **minimitzar** els kilòmetres de buit dels nostres recursos, arribar amb puntualitat als clients i complir amb les restriccions normatives relacionades amb conductors i cabotatges.

El problema que intentem resoldre ve donat per bastants factors com poden ser: la complexitat dels càlculs d'emparellament, la falta de visió global de tots els recursos amb les restriccions de cadascun, la visió global de la disponibilitat dels xofers, etc.

Tot això fa que l'assignació de recursos a viatges corresponents sigui realment complexa i actualment està lluny de ser la més òptima possible.

També s'ha de tenir en compte que l'any passat (2015) disposàvem de 300 tractores i 300 remolcs però actualment ja disposem de 370 tractores i vora 400 remolcs i aquestes xifres

van en augment. Per tant, aquests càlculs cada cop són més complexos i requereixen més temps i persones qualificades.

El factor decisiu per engegar aquest projecte ha estat el fet que a partir d'ara, en certes campanyes, treballarem d'una nova forma que genera una gran càrrega d'assignació, desassignació i replanificació dels recursos.

L'objectiu principal d'aquest projecte és fer aquesta planificació de la forma més òptima i automàtica possible.

1.5 Estructura del document

- Capítol 1: Es descriu el problema i els objectius del treball, es mostren les metodologies bàsiques per a la planificació de recursos i es presenta l'estat de l'art en la matèria.
- Capítol 2: Es descriu el model matemàtic de referència i la seva adaptació al nostre cas concret. També es descriuen els mòduls que formen part del projecte i les decisions més importants que s'han pres per cadascun.
- Capítol 3: Es fa la validació del model, s'expliquen els procediments que s'han dut a terme per fer-ho i es mostra un anàlisis de rendiment dels diferents mòduls.
- Capítol 4: Es descriuen les diferents fases del procés d'implantació d'aquest projecte en l'empresa.
- Capítol 5: Es detallen varis punts a implementar i millorar en futures extensions del projecte.
- Capítol 6: Es comenten les conclusions del projecte.
- Capítol 7: Es mostra la bibliografia i les fonts consultades.

1.6 Tecnologies relacionades

Actualment existeix una gran varietat de software per solucionar diferents tipus de problemes d'optimització. En aquest apartat es comentarà quin s'ha utilitzat per validar i resoldre el model presentat en el present treball. També es mencionaran alternatives gratuïtes i propietàries a aquest software i es mostraran els formats que existeixen per modelar els problemes d'optimització.

Finalment, també es mencionaran les tecnologies més importants que s'han fet servir en aquest projecte.

1.6.1 Tecnologies usades

1.6.1.1 IBM Ilog CPLEX optimizer

Per implementar el mòdul d'optimització s'ha utilitzat el software CPLEX (propietat de IBM) amb una llicència d'estudiant. L'he escollit perquè és un dels més utilitzats a nivell mundial i té un rendiment excel·lent per a problemes relativament grans. Usaré CPLEX per avaluar el model inicial i totes les posteriors modificacions.

Aquest software disposa d'una suite molt completa des d'on programar els models amb un llenguatge propi, anomenat OPL (Optimization Programming Language).

En la Figura 1 es mostra un exemple de modelatge en llenguatge OPL.

```
78  dvar interval a[Enters] size 1;
79
80  // each sector is modelled by a resource
81  cumulFunction r[i in SectorNames] = sum(en in Enters : e[en].sector == i) pulse(a[en], 1);
82
83  execute {
84          cp.param.FailLimit = 20000;
85  }
86
87  dexpr int totalDelay = sum(i in Flights) delay[i];
88
89  minimize totalDelay;
90  constraints {
91
92    // the capacity rate is adapted to intervals of 10 minutes;
93    // the time scale of a resource is divided by the time step
94    forall (i in SectorNames)
95        forall (p in periods[i])
96           alwaysIn(r[i], (p.start.hours * 60 + p.start.minutes) div timeStep,
97                          (p.end.hours * 60 + p.end.minutes) div timeStep,
98                             0,
99                          (p.rate * timeStep + 59) div 60);
100
101
102    // a flight enters a sector at its expected time-over plus its delay;
103    // since the time scale of a resource is divided by the time step,
104    // we do the same for the start time of the activity
105    forall (i in Enters)
106       startOf(a[i]) == (delay[e[i].flight] + e[i].eto.hours * 60 + e[i].eto.minutes) div timeStep;
107
108    forall(i in SectorNames)
109      r[i] <= nbOfFlights;
110 }
111
112 execute {
113   writeln("total delay = " + totalDelay);
114 }
```

Figura 1. Llenguatge OPL

CPLEX també disposa d'una interfície per ser cridat des de llenguatges com C/C++, .NET, Java, Python o inclús Matlab. Per comoditat i per fer les validacions del model sense invertir temps en aprendre el llenguatge OPL, programaré i executaré el model des de Java.

Segons el propi manual del software, aquest és capaç d'escollir diferents algoritmes depenent del tipus de problema per disminuir el temps d'execució. Aquests algorismes es descriuen a continuació:

- Automàtic
- Primal Símplex
- Dual Símplex
- Network Símplex
- Barrier
- Sifting
- Concurrent

Si es tria el mètode automàtic, CPLEX seleccionarà l'algoritme que hauria de funcionar millor pel tipus de problema a resoldre.

Actualment, el comportament del **mode automàtic** és que quasi sempre es farà servir l'algoritme Dual Símplex pels problemes lineals que s'intenten resoldre des de zero. Quan es continua una optimització des d'un estat avançat, el programa mirarà si és millor fer servir el mode Dual Símplex o el mode Primal Símplex.

Si s'han especificat múltiples fils d'execució, s'ha escollit el mètode automàtic i el problema és de programació lineal, el programa escollirà el **mode Concurrent**.

Quan hi ha disponibles tres o més fils d'execució i es selecciona el mode Concurrent poden executar-se dos modes diferents:

El Mode oportunista que fa servir l'algoritme Dual Símplex en un fil, el Primal Símplex en un segon fil i el Parallel Barrier a la resta de fils disponibles. El **Mode determinista** que fa servir l'algoritme Dual Símplex en un fil i el Parallel Barrier a la resta.

1.6.1.2 Microsoft Dynamics Axapta 2009

Es fa servir un sistema ERP per la gestió de l'empresa. Des de l'any 2005 s'utilitza Microsoft Dynamics Axapta 2009 ja que és un sistema que suporta un entorn multi empresa i multi divisa.

Axapta 2009 és un software de planificació de recursos empresarials (ERP) propietat de Microsoft. Disposa d'un entorn de desenvolupament integrat anomenat MorphX que conté varies eines com un depurador, un analitzador de codi i una interfície de consulta de la base de dades. Es programa amb un llenguatge propietari anomenat X++. Aquest llenguatge està orientat a objectes i té similituds amb el C#. No existeix cap tipus d'aritmètica de punters i els objectes es creen amb l'operador "new".

X++ està provist de sintaxis per fer crides directes a .NET. Actualment existeixen noves versions d'Axapta (la última és Dynamics AX7) que estan integrades en l'entorn de programació de Visual Studio i milloren en gran mesura la productivitat dels desenvolupadors.

1.6.1.3 Java

Java és un llenguatge de programació que està en funcionament des de l'any 1995. És un llenguatge orientat a objectes i es va dissenyar per a que fos senzill. Necessita d'un intèrpret per a ser executat. També cal dir que està altament tipificat, cosa que li dona una gran robustesa.

Una de les característiques més importants es que és un llenguatge portable, d'alt nivell i independent de la plataforma on s'ha d'executar.

En un projecte d'aquest tipus on el rendiment i l'aprofitament dels recursos informàtics és primordial, es pot fer estrany veure que s'ha programat en Java. Normalment s'hauria fet servir un llenguatge com el C++, que també està orientat a objectes, és més ràpid i permet gestionar més eficientment els recursos. Si s'ha utilitzat Java en comptes de C++ és pel gran nivell de productivitat del que disposa i del temps limitat del projecte.

1.6.1.4 .NET 3.5

.NET és una plataforma de desenvolupament i execució d'aplicacions. Disposa de totes les eines i serveis que es necessiten per desenvolupar aplicacions empresarials però també proporciona mecanismes robustos i eficients per assegurar que l'execució de les mateixes sigui òptima. Aquesta plataforma inclou:

- Un entorn d'execució d'aplicacions.
- Un conjunt de biblioteques i funcionalitats reutilitzables.
- Un conjunt de llenguatges de programació d'alt nivell amb els seus compiladors.
- Utilitats i eines per simplificar les tasques comunes de desenvolupament.
- Documentació i guies d'arquitectura que descriuen les millors pràctiques de disseny.

En aquest projecte s'ha usat la versió 3.5 d'aquesta plataforma. La data pública de llançament d'aquesta versió va ser el 19 de novembre de 2007.

1.6.1.5 Protocol HTTP

És un protocol de comunicació que permet la transferència de dades. No té estat, per tant no guarda informació sobre connexions anteriors. Està orientat a transaccions i segueix l'esquema de petició-resposta entre un client i un servidor.

Els missatges de HTTP són en text pla, el que simplifica la seva lectura i depuració, tot i que això significa que els missatges són més llargs.

Aquest protocol disposa d'una sèrie de mètodes de petició, també anomenats verbs. Els més importants i utilitzats són els següents:

- GET: Demana una representació del recurs especificat.
- POST: Envia dades per a que siguin processades per al recurs especificat.
- PUT: Puja o carga un recurs especificat.
- DELETE: Elimina un recurs especificat.
- HEAD: És com un GET però la resposta no retorna el cos de la petició. Útil per recuperar metadades dels encapçalaments de la resposta.

1.6.1.6 Spring Framework

Spring Framework proporciona un model complet de programació i configuració per aplicacions empresarials basades en Java. Un dels punts més importants és que Spring es centra en realitzar tota la part "pesada" de configuració dels projectes, d'aquesta manera els desenvolupadors es poden centrar en el disseny i implementació de l'aplicació i no han d'invertir temps en fer configuracions específiques per a diferents entorns.

1.6.1.7 Maven

El principal objectiu de Maven [15] és permetre a un desenvolupador comprendre l'estat complet d'un esforç de desenvolupament en el menor període de temps. Per aconseguir això Maven aborda diferents àrees:

- Fer que el procés de construcció (build) sigui fàcil.
- Proporcionar un sistema de construcció uniforme.
- Proporcionar informació de qualitat sobre el projecte.
- Proporcionar ajuda i mostrant directrius sobre les millors pràctiques de desenvolupament
- Permetre una migració transparent a noves característiques.

1.6.1.8 JSON

JSON és un format de text per a l'intercanvi de dades. Va néixer com una alternativa a XML i en els últims anys ha crescut de forma considerable, principalment degut a que és un format natiu en el llenguatge de programació Javascript. Aquest format pot ser llegit per qualsevol llenguatge de programació i per tant pot ser usat per intercanviar informació entre diferents tecnologies (exactament pel motiu que s'ha usat en aquest projecte, combinat amb el protocol HTTP). Un exemple d'aquest format:

```
{
    "type": 1,
    "name": "PBL-019314",
    "latitude": 49.2717,
    "longitude": -1.0149,
    "carrega": 0,
    "id": 8,
    "tiempoIni": 7888854,
    "tiempoFin": 7888974
}
```


Figura 2. Exemple format JSON

1.6.1.9 Bing Maps

Bing Maps és un servei de mapes online propietat de Microsoft. Disposa d'un gran nombre de característiques de visualització i d'una API per a fer consultes des d'aplicacions externes. En aquest projecte s'ha usat el servei REST per trobar les distàncies entre diferents coordenades. Dins d'aquest servei es disposa de cinc tipus de crides per fer diferents consultes, aquestes són:

- Locations API: Per treure direccions a partir de coordenades i viceversa.
- Elevations API: Per treballar amb informació sobre elevacions del terreny.
- Imagery API: Per extreure diferents tipus d'imatges del mapa.
- Routes API: Per extreure informació de rutes i distàncies.
- Traffic API: Retorna informació sobre incidents de trànsit i problemes varis.

1.6.1.10 MySql

És una base de dades SQL. Actualment és una de les més utilitzades a nivell mundial i disposa d'una gran quantitat de documentació. Compta amb un nombre elevat de complements que augmenten la seva utilitat, rendiment i fiabilitat.

1.6.2 Alternatives a CPLEX

Avui en dia existeix un gran nombre d'alternatives a CPLEX. En l'àmbit empresarial hi ha disponibles des de grans Frameworks fins a petits optimitzadors que només serveixen per un tipus de problema en concret. En l'àmbit de codi lliure també existeixen grans solucions però s'ha de dir que no s'acosten al nivell de rendiment de CPLEX.

En la Taula 1 es mostra un resum d'algunes alternatives gratuïtes a CPLEX.

Software	Llicència	Observacions
ADMB	BSD	Framework d'optimització no lineal.
APOPT	GPL	Framework de programació entera mixta no lineal.
COIN-OR SIMPHONY	Eclipse v.1	Programació entera.
GLPK	GPL	Es un pack encarat a resoldre grans problemes de programació lineal i programació entera mixta entre d'altres.
IPOPT	CPL	Optimització de gran escala per a sistemes continus no lineals
MIDACO	BY-NC-ND	Framework d'optimització entera mixta no lineal

OpenMDAO	ASL	Framework d'optimització, anàlisis i disseny multidisciplinari
SCIP	ZIB Academic License	Programació entera mixta i entera mixta no lineal. Framework per a programació entera amb restriccions.

Taula 1. Alternatives a CPLEX de codi lliure

La Taula 2 mostra les alternatives de software propietari.

Software	Observacions
APMonitor	Suite d'optimització de problemes a gran escala
Artelys Knitro	Suite d'optimització de problemes a gran escala
BARON	Programació no lineal i entera mixta
FortMP	Programació entera, lineal i quadràtica
FortSP	Programació estocàstica
Gurobi	Programació entera, lineal i quadràtica
Kimeme	Plataforma per a problemes multi objectiu i disseny multi disciplinari
modeFRONTIER	Plataforma d'integració per problemes multi objectiu i optimització multi disciplinària
Maple(software)	Programació lineal, quadràtica, no lineal, continua i entera
MATLAB	Programació lineal, entera, quadràtica no lineal, etc.
WOLFRAM Mathematica	Problemes de gran escala de programació lineal, no lineal, contínua i entera
MOSEK	Programació lineal, quadràtica, cònica i convexa no lineal, contínua i entera
NAG	Programació lineal, quadràtica, no lineal, continua i entera
NMath	Programació lineal, quadràtica i no lineal
WORHP	Programació amb gran volum de dades de problemes continus no lineals
XPRESS	Programació lineal, entera, quadràtica i no lineal

Taula 2. Alternatives a CPLEX de codi propietari

No totes les alternatives a CPLEX mencionades a les taules 1 i 2 són vàlides per la resolució del model plantejat en aquest projecte, de fet, de les alternatives gratuïtes considero que SCIP, MIDACO i GLPK són les úniques viables, i d'aquestes, SCIP és la millor. Aquests Frameworks poden manejar desenes de milers de variables simultàniament, dins dels sistemes gratuïts són dels mes ràpids i disposen d'eines per ser cridats des de diferents llenguatges de programació o de softwares com Excel i Matlab entre d'altres.

Considero que SCIP és la millor opció perquè la suite que incorpora és molt completa, està desenvolupat en ANSI C i segons la web oficial [19] és compatible amb les versions de 32 i 64 bits de Linux, Mac, Windows, SunOS i Android. Finalment, a la mateixa web oficial també es mostra com SCIP, de mitjana, és la alternativa gratuïta més ràpida.

Pel que fa a les alternatives de software propietari, no he pogut comparar en profunditat totes les característiques de cadascuna d'elles per escollir la millor, però he anat filtrant les diferents suites i Frameworks segons les característiques que els limitaven per dur a terme aquest projecte. Al final, les opcions que m'han semblat més completes han estat Gurobi i WOLFRAM Mathematica (a part de CPLEX).

Gurobi és una suite potentíssima, el nucli d'optimització de la suite és el més ràpid del mercat en la major part de tests i pot resoldre problemes de programació lineal, quadràtica, entera mixta lineal i entera mixta quadràtica. Està desenvolupat per aprofitar el paral·lelisme dels sistemes informàtics, disposa d'avançades heurístiques, implementa una gran gama d'APIs per simplificar la corba d'aprenentatge en la programació dels models matemàtics, és compatible amb diferents llenguatges/softwares i moltes més funcionalitats prometedores que es poden consultar a la web oficial del software [20].

Per altra banda, **Mathematica** encara disposa d'una suite més completa, que a part de permetre optimitzar els problemes "personalitzats" que li puguem introduir, també disposa d'una gran varietat de mòduls especialitzats en matèries com poden ser: el sector aeroespacial i de defensa, tot tipus d'enginyeries, finances, estadística, anàlisis empresarial, astronomia, etc. Aquests mòduls faciliten en gran mesura les tasques d'optimització i estalvien la pesada feina de dissenyar i implementar models específics en aquests àmbits. També cal dir que la documentació d'aquest Framework és excepcional ja que dins la web oficial [21], existeixen apartats amb documentació, llibres, seminaris virtuals, videotutorials i altres documents interesants per formar-se en l'entorn de forma ràpida.

En resum, Mathematica crec que és una suite sobredimensionada (per al present projecte) i no s'aprofitarien la major part de les seves característiques. Gurobi no el vaig escollir perquè a l'inici del projecte no vaig indagar prou en totes les seves possibilitats, però ara, veient tot el que incorpora i que també disposa de llicències gratuïtes per a estudiants, és possible que m'hagués decantat per aquesta opció. Respecte a SCIP puc dir que vaig descarregar-me la suite i vaig intentar fer proves amb models senzills però contínuament tenia problemes en la instal·lació del software cosa que em va impedir seguir amb les proves. Finalment, vaig decidir usar CPLEX perquè és una de les suites més utilitzades a nivell mundial, vaig trobar gran quantitat de documentació tant oficial com en diferents fòrums, també és una de les que té millor rendiment i perquè van ser els primers en atorgar-me una llicència gratuïta per a estudiants. Gràcies a aquesta rapidesa i a la facilitat de configuració de la suite, em va ser molt fàcil començar a fer les primeres proves amb models senzills i per tant ja em vaig quedar amb aquesta plataforma.

1.6.3 Formats de modelització

Existeixen diferents formats per representar models matemàtics d'optimització, d'aquesta forma poden ser processats pels softwares esmentats anteriorment.

En la següent llista es poden veure els més utilitzats actualment:

- AMPL: A mathematical Programming Language. Actualment és el llenguatge més popular amb diferència. Va ser creat el 1985 i te un estil de programació mixt declaratiu-imperatiu.
- GAMS: General Algebraic Modeling System. És el segon llenguatge més popular. Va ser creat el 1976 i ha evolucionat molt des de llavors.
- LP: Usat en problemes de programació lineal. Molt intuïtiu.
- CPLEX (OPL): Llenguatge creat per IBM i natiu del software CPLEX Suite Studio.
- SPARSE SDPA: Format pensat per usar amb algoritmes SDPA i variants.

- MPS: Tots els softwares d'avui en dia accepten aquest format tot i que ja està molt desfasat i és molt difícil d'entendre. Està pensat per ser interpretat per màquines que funcionen amb targetes perforades.
- SMPS: Format pensat per programació estocàstica.

En aquest projecte no s'ha fet servir cap d'aquests llenguatges per a la modelització dels problemes ja que aquests s'han modelat des de la interfície que comunica JAVA amb la suite d'optimització CPLEX.

D'altra banda, durant el desenvolupament del projecte he fet servir el **llenguatge LP** per exportar els diferents models que he anat testejant. L'he escollit perquè m'ha facilitat en gran mesura la interpretació dels models i sobretot m'ha permès identificar de forma molt ràpida els errors de programació tant en la funció objectiu com en les restriccions.

Finalment, deixant de banda el format MPS que està completament desfasat, la resta de formats no són millors o pitjors entre ells en general. Cada format té les seves particularitats que el fan més útil de cara a un tipus de problema o a un altre. Per exemple, el SMPS està dissenyat per modelar problemes estocàstics o el LP per fer-lo servir en problemes de programació lineal.

2 Desenvolupament

2.1 Anàlisis del model inicial

A l'inici d'aquest projecte no tenia cap tipus d'experiència en el món de l'optimització, i per tant vaig haver de buscar gran quantitat de bibliografia per ficar-me al dia i veure com es trobava l'estat de l'art en la matèria. Finalment, després d'estudiar bastants models vaig decantar-me per analitzar i entendre el model proposat per Pairoj et al. en [1].

En la tesi esmentada es descriu un model per resoldre problemes del tipus "Multiple Depot Pickup Delivery Problem". Aquest model permet optimitzar flotes heterogènies de vehicles, localitzades en diferents punts de partida. Cada vehicle ha d'anar a carregar i posteriorment descarregar en una localització diferent. Tant la càrrega com la descarrega han de ser servides pel mateix vehicle. El nombre de vehicles estacionat a cada punt de partida és conegut. Tant les càrregues com les descàrregues també són conegudes abans de començar el viatge. Els vehicles han de sortir d'un magatzem inicial i acabar en un magatzem final. Cada vehicle té una capacitat limitada i una distància màxima per recórrer. **L'objectiu** de la funció és **trobar la distància mínima total** per satisfer totes les ordes dels clients.

La notació per aquest problema és la següent:

- n és el nombre d'ordres dels clients.
- m és el nombre de vehicles disponibles.
- El problema es representa en un graf.
- $P = \{1, ..., n\}$ són els nodes de càrrega.
- $D = \{n + 1, ..., 2n\}$ són els nodes de descàrrega.
- La ordre i es representa pels nodes i i $i+n$.
- K és el conjunt de tots els vehicles. $|K| = m$.
- $N = P \cup D$.
- $T_k = 2n + k$, $k \in K$ representa els magatzems de sortida del vehicle k.
- $T'_k = 2n + m + k$, $k \in K$ representa els magatzems d'arribada del vehicle k.
- $G = (V, A)$ és un graf que conté:
  - Els nodes $V = N \cup \{T_1, ..., T_k\} \cup \{T'_1, ..., T'_m\}$.
  - Els arcs $A = V \times V$.
- Per cada arc $(i,j) \in A$:
  - La distància $d_{ij} > 0$.
  - El temps de viatge $t_{ij} > 0$.
- Es compleix la següent desigualtat pel temps:
  - $t_{ij} < t_{il} + t_{lj} \; \forall \; i,j,l \in V$.
- Per a cada node $i \in N$, $l_i > 0$, $i \in P$ és el nombre de palets a carregar al node i.
- La capacitat del vehicle $k \in K$ es denotada com C_k.
- R^k es la distància màxima permesa pel vehicle k.

En aquest model hi ha quatre tipus de variables de decisió:

- X_{ijk} $i, j \in V, k \in K$ és una variable binària que el seu valor és 1 si l'arc entre i i j es fa servir pel vehicle k i el valor és 0 si no es fa servir.
- S_{ik} $i \in V, k \in K$ és un enter positiu que indica quan un vehicle k comença un servei al node i.
- L_{ik} $i \in V, k \in K$ és un enter positiu corresponent al total de la carga que porta al vehicle k al vèrtex i.
- Z_i i $\in P$ és una variable binària que indica si la càrrega i s'ha assignat a algun vehicle o no.

Per raons pràctiques, s'ha reduït el conjunt d'arcs del graf A al següent conjunt d'arcs A':

$A' = \{(i,j) : i, j \in V, i \neq T'_k, j \neq T_k, i \neq j, i \in P \rightarrow j \neq T'_k, i = T_k \rightarrow j \notin D, i \in D \rightarrow j \notin P \text{ on } i = j + n\}$

Això vol dir que en lloc d'agafar la combinació de tots els possibles arcs del graf, només es tindran en compte els arcs que són representatius en els càlculs de la resolució del problema. Per exemple, els arcs que van d'un punt d'inici a una càrrega, d'una càrrega a una descàrrega, o d'una descàrrega a un punt final. En canvi es descarten els arcs que poden anar d'un punt d'inici a un punt final directament, d'una càrrega a un punt final, etc.

La funció objectiu del model descrit és la següent:

$$Min\ \alpha \sum_{k \in K} \sum_{(i,j) \in A'} d_{ij} X_{ijk} + \gamma \sum_{i \in P} Z_i \qquad (3.1)$$

Subjecte a les restriccions:

$$\sum_{k \in K} \sum_{j:(i,j) \in A'} X_{ijk} + Z_i = 1 \qquad \forall i \in P \qquad (3.2)$$

$$\sum_{j:(i,j) \in A'} X_{ijk} - \sum_{j:(n+i,j) \in A'} X_{n+i,j,k} = 0 \qquad \forall k \in K, i \in P \qquad (3.3)$$

$$\sum_{j \in P \cup \{T'_k\}} X_{T_k,j,k} = 1 \qquad \forall k \in K \qquad (3.4)$$

$$\sum_{i \in D \cup \{T_k\}} X_{i,T',k} = 1 \qquad \forall k \in K \qquad (3.5)$$

$$\sum_{i:(i,j) \in A'} X_{ijk} - \sum_{i:(i,j) \in A'} X_{jik} = 0 \qquad \forall k \in K, \forall j \in N \qquad (3.6)$$

$$S_{ik} + s_i + t_{ij} - M(1 - X_{ijk}) \leq S_{jk} \qquad \forall k \in K, \forall (i,j) \in A' \qquad (3.7)$$

$$S_{ik} \leq S_{n+i,k} \qquad \forall k \in K, \forall i \in P \qquad (3.8)$$

$$L_{ik} + l_j - M(1 - X_{ijk}) \leq L_{jk} \qquad \forall k \in K, \forall (i,j) \in A' \qquad (3.9)$$

$$Max\{0, l_j\} \leq L_{ik} \leq Min\{C_k, C_k + l_j\} \qquad \forall k \in K, \forall i: (i,j) \in A' \qquad (3.10)$$

$$\sum_{(i,j)\in A'} d_{ij} X_{ijk} \leq R^k \qquad \forall k \in K \qquad (3.11)$$

$$X_{ijk} \in \{0,1\} \qquad \forall k \in K, \forall (i,j) \in A'$$

$$Z_i \in \{0,1\} \qquad \forall i \in P$$

$$S_{ik} \geq 0 \qquad \forall k \in K, \forall i \in V$$

$$L_{ik} \geq 0 \qquad \forall k \in K, \forall i \in V$$

L'objectiu principal (3.1) és minimitzar la suma ponderada de la distància recorreguda i el nombre d'ordres no despatxades.

La restricció (3.2) assegura que si una ordre no es despatxa es marca com a no despatxada. La restricció (3.3) assegura que tant la càrrega com la descàrrega corresponent són despatxades pel mateix vehicle. Les restriccions (3.4) i (3.5) asseguren que cada vehicle comença a un magatzem inicial i acaben al magatzem final designat. Amb la restricció (3.6) ens assegurem que es construeixen camins consecutius entre T_k i T'_k. La restricció (3.7) assegura que S_{ik} s'estableix correctament durant els camins. La restricció (3.8) assegura que cada càrrega precedeix a la descàrrega corresponent. Les restriccions (3.9) i (3.10) asseguren que en cada moment es satisfà la càrrega màxima de cada vehicle. La restricció (3.11) assegura que cada vehicle no recorre més distància de la distància màxima que se li permet.

En un primer moment era raonable començar a usar aquest algoritme ja que feia servir un model àmpliament comprovat en diferents tesis i semblava bastant compatible amb una part simplificada de la nostra lògica de negoci. Posteriorment, vaig detectar que es podien canviar i ometre bastantes restriccions per alleugerar la complexitat del càlcul i els recursos computacionals necessaris. També vaig creure convenient afegir nous paràmetres a la funció objectiu. Al següent apartat es detalla aquest nou model matemàtic.

2.2 Proposta d'un nou model

El model matemàtic MDPDP és un bon model teòric però com he comentat anteriorment, cada empresa pot arribar a tenir moltes particularitats que fan que aquests models teòrics no compleixin les expectatives reals de l'empresa.

El primer que s'ha fet per simplificar el problema ha estat fusionar tots els punts de cada ordre per a que només hi hagi una càrrega i una descàrrega. En la Figura 3 es pot veure aquesta simplificació:

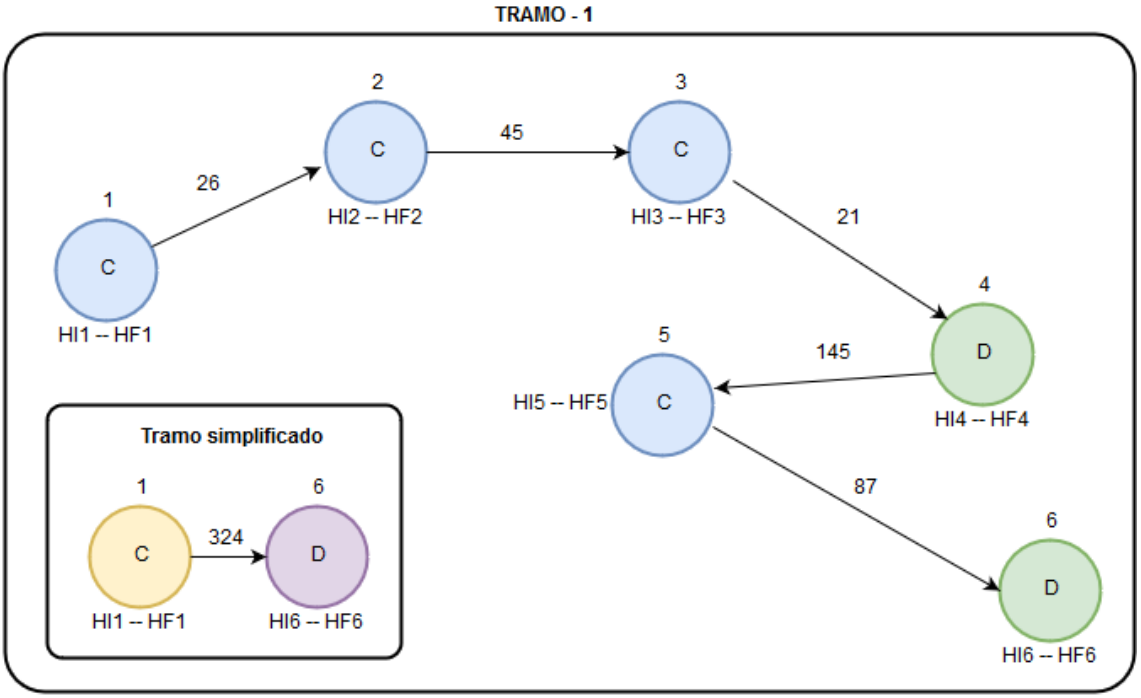


Figura 3. Simplificació de trams

La imatge representa el que podria ser perfectament una ordre d'un client, aquesta ordre tindria quatre càrregues i dues descàrregues intercalades. Cada node tindria una finestra de temps a la qual s'ha d'arribar amb el vehicle i entre cada node s'ha de recórrer una certa distància. En el tram simplificat s'han eliminat els nodes intermedis i s'ha sumat les distàncies entre ells per acabar amb una sola parella de càrrega-descàrrega.

Tal com funciona la nostra empresa i diferint del model MDPDP, no es disposa d'uns magatzems de sortida i d'arribada dels quals s'hagi de sortir o d'arribar obligatòriament a l'acabar amb una ordre. Normalment es dona el cas que el punt de sortida per a la ordre actual d'un vehicle és una àrea de descans, un pàrquing de seguretat, la última descàrrega de la ordre del client anterior, etc. Els punts d'arribada també poden ser els acabats d'esmentar.

Es podria donar el cas que el lloc de finalització fos el mateix que l'inicial, o que més d'un vehicle tingués els mateixos inicis i destins finals, però el problema s'ha definit com a que cada vehicle té un inici i un destí diferents.

També utilitzem altres modalitats de treball que sí disposen de magatzems de sortida i d'arribada als quals s'ha de sortir o arribar obligatòriament, però no és el model que s'estudia en aquest projecte.

Així doncs, la definició gràfica de la ruta que seguirà un vehicle que ha estat assignat a una ordre és la següent (Figura 4):

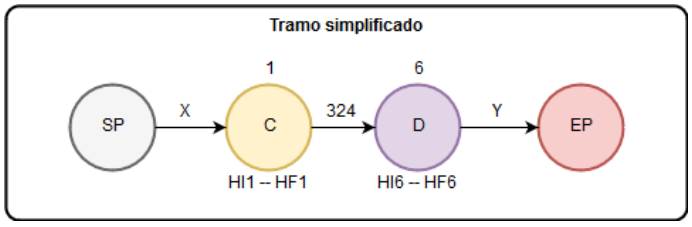


Figura 4. Ruta en format de graf dirigit

SP es refereix al "Starting Depot" o magatzem inicial, EP es refereix a "End Depot" o magatzem final, la variable X són els **kilòmetres de buit** des de l'origen del vehicle fins a la primera càrrega de la ordre i la variable Y són els kilòmetres des de la última descàrrega de la ordre fins al lloc de descans (o fins la primera càrrega de la següent ordre).

2.2.1 Descripció

El nou model consta d'una flota de vehicles homogènia. Cada vehicle surt d'un emplaçament de sortida particular i ha d'acabar a un emplaçament d'arribada particular. En una ordre la càrrega sempre ha de precedir a la descàrrega i les dos han de ser despatxades pel mateix vehicle. Es coneix el nombre total de vehicles i el nombre total d'ordres dels clients. **No** es té en compte la capacitat màxima del vehicle ja que d'això se'n encarrega el client al demanar la ordre i per tant aquest sap que cada vehicle pot portar un determinat nombre de palets. Hi ha una distància màxima que cada vehicle pot recórrer.

La funció objectiu canvia respecte al model MDPDP afegint nous paràmetres a la fórmula. El primer és el consum per kilòmetre (en litres) de cada tractora, d'aquesta manera, en igualtat de condicions, l'algoritme emparellarà les rutes més llargues als vehicles que consumeixen menys. També s'ha afegit un paràmetre per destacar la prioritat particular de cada ordre (definida segons criteris de l'empresa), d'aquesta forma, en igualtat de condicions, es despatxaran abans les ordres més prioritàries. Finalment, l'últim paràmetre que he afegit és el benefici net previst que comporta despatxar cada ordre. En igualtat de condicions, es despatxaran primer les ordres que comporten més benefici per l'empresa.

En resum, amb la nova funció objectiu, el valor resultant no optimitzarà el nombre mínim de kilòmetres que s'acabarien recorrent sinó que serà un nombre que provindrà del producte dels diferents criteris esmentats. Per tant, **l'objectiu** d'aquest model serà **minimitzar el cost total**.

Abans de mostrar la notació d'aquest model es convenient fer un petit aclariment. Normalment els models de problemes d'optimització estan pensats per ser programats fent servir matrius de dades i de variables. En el meu cas, tal i com s'explicarà amb detall a l'apartat 2.3.7 d'aquest document, he aprofitat els avantatges de la programació orientada a objectes i per tant la representació i notació matemàtica incorpora canvis respecte al model MDPDP.

La notació per aquest model és la següent:

- n és el nombre d'ordres dels clients.
- m és el nombre de vehicles disponibles.
- $G = (V, A)$ és el graf on es representa el problema.
    - V és el conjunt de tots els nodes del graf.
    - $A = V \times V$ són els arcs del graf.
    - $C \subset V$ és el subconjunt de nodes de càrrega del graf.
    - $D \subset V$ és el subconjunt de nodes de descàrrega del graf.
    - $S \subset V$ és el subconjunt de punts de partida del graf.
    - $E \subset V$ és el subconjunt de punts d'arribada del graf.
- T és el conjunt d'ordres dels clients. $|T| = n$.
    - $\forall$ c, $\exists!(c, d) \mid c \in C, d \in D$. Per a cada càrrega existeix una sola parella càrrega-descàrrega.
- K és el conjunt de totes els vehicles. $|K| = m$.
- Per a cada arc $(i,j) \in A$:
    - La distància $d_{ij} > 0$.

- El temps de viatge $t_{ij} > 0$.

- R^k és la distància màxima permesa pel vehicle $k \in K$.
- L^k és el consum en litres per cada kilòmetre del vehicle $k \in K$.
- No es contempla la capacitat màxima de cada vehicle ni el nombre de palets a carregar /descarregar a cada node.
- b_i és el benefici previst de despatxar la ordre $i \in T$

En aquest model hi ha dos tipus de variables de decisió:

- X_{ijk} $i, j \in V, k \in K$ és una variable binària que el seu valor és 1 si l'arc entre i i j es fa servir pel vehicle k i el valor és 0 si no es fa servir.
- Z_i i $\in C$ és una variable binària que indica si la càrrega i s'ha assignat a algun vehicle o no.

La funció objectiu d'aquest nou model és:

$$Min\ \alpha \sum_{k \in \mathrm{K}} \sum_{(i,j) \in \mathrm{A}\prime} d_{ij} L^k X_{ijk} + \gamma \sum_{i \in \mathrm{C}} b_i Z_i \qquad (4.1)$$

Subjecte a les restriccions:

$$\sum_{j:(s,c) \in \mathrm{A}\prime} X_{sck} - \sum_{j:(c,d) \in \mathrm{A}\prime} X_{cdk} = 0 \qquad \mathrm{c} \in \mathrm{C}, \mathrm{d} \in \mathrm{D}, \mathrm{s} \in \mathrm{S}, \forall \mathrm{k} \in \mathrm{K} \qquad (4.2)$$

$$\sum_{j:(c,d) \in \mathrm{A}\prime} X_{cdk} - \sum_{j:(d,e) \in \mathrm{A}\prime} X_{dek} = 0 \qquad \mathrm{c} \in \mathrm{C}, \mathrm{d} \in \mathrm{D}, \mathrm{e} \in \mathrm{E}, \forall \mathrm{k} \in \mathrm{K} \qquad (4.3)$$

$$\sum_{j:(s,c) \in \mathrm{A}\prime} X_{sck} - \sum_{j:(d,e) \in \mathrm{A}\prime} X_{dek} = 0 \qquad \mathrm{c} \in \mathrm{C}, \mathrm{d} \in \mathrm{D}, \mathrm{s} \in \mathrm{S}, \mathrm{e} \in \mathrm{E}, \forall \mathrm{k} \in \mathrm{K} \qquad (4.4)$$

$$\sum_{\mathrm{k} \in \mathrm{K}} X_{cdk} <= 1 \qquad \forall \mathrm{c} \in \mathrm{C}, \mathrm{d} \in \mathrm{D} \qquad (4.5)$$

$$\sum_{i:(s,c) \in \mathrm{A}\prime} X_{sck} <= 1 \qquad \forall \mathrm{k} \in \mathrm{K}, \mathrm{s} \in \mathrm{S}, \mathrm{c} \in \mathrm{C} \qquad (4.6)$$

$$\sum_{(i,j) \in \mathrm{A}\prime} d_{ij} X_{ijk} \leq R^k \qquad \forall \mathrm{k} \in \mathrm{K} \qquad (4.7)$$

$$\sum_{k \in \mathrm{K}} \sum_{j:(i,j) \in \mathrm{A}\prime} X_{ijk} + Z_i = 1 \qquad \forall \mathrm{i} \in \mathrm{P} \qquad (4.10)$$

$$X_{ijk} \in \{0,1\} \qquad \forall \mathrm{k} \in \mathrm{K}, \forall (\mathrm{i}, \mathrm{j}) \in \mathrm{A}'$$

$$Z_i \in \{0,1\} \qquad \forall \mathrm{i} \in \mathrm{P}$$

L'objectiu principal (4.1) és minimitzar la suma ponderada del cost de la distància recorreguda i el nombre d'ordres no despatxades.

La restricció (4.2) assegura que si un vehicle va del seu punt de partida fins a una carga, llavors haurà d'anar obligatòriament a la descàrrega corresponent. La (4.3) diu que si un vehicle va d'una càrrega a una descàrrega, llavors aquest haurà d'anar a un punt final. La restricció (4.4) diu que si un vehicle surt des del seu punt d'inici, el mateix vehicle haurà d'anar al seu punt final. La restricció (4.5) assegura que cada càrrega (o cada ordre) només podrà ser feta com a molt per un vehicle (pot ser que no es despatxi la ordre, amb la penalització corresponent). La (4.6) serveix per "engegar" un vehicle i fer-lo sortir de la seva base, també pot ser que el vehicle no s'emparelli amb cap ordre i per tant no s'engegui. La restricció (4.7) assegura que cap vehicle superarà la seva distància màxima permesa. Amb la restricció (4.8) ens assegurem que si una ordre no es despatxa, s'apliqui la corresponent penalització.

2.3 Programació dels diferents mòduls del projecte

Abans de començar a definir com s'ha implementat el projecte és necessari saber el que s'inclou dins del mateix. Aquest apartat es dividirà en les diferents àrees que intervenen en el procés d'optimització:

1. ERP: El sistema de gestió que fem servir a l'empresa i des d'on s'extreuen les dades que es volen optimitzar. El sistema és un Microsoft Dynamics Axapta 2009.
2. Llibreria .NET: Degut a l'antiguitat de l'ERP fa falta la implementació d'aquesta llibreria per preparar les dades i fer d'interfície entre l'ERP i el servei web d'optimització.
3. Servei de càlcul de distàncies: S'ha usat Bing Maps per extreure les distàncies entre coordenades.
4. Taula de distàncies: Per reduir el temps de consulta de les distàncies, s'ha implementat una taula de distàncies usades anteriorment en una base de dades MySQL.
5. Servei REST: S'ha implementat un servei REST usant el Framework Spring per poder cridar el servei d'optimització de forma més transparent i sense haver de tenir en compte dependències especials.
6. Optimitzador: L'executable programat en Java que es cridat pel servei REST. Optimitza els recursos i retorna el resultat al servidor web.

En la Figura 5 es poden veure aquests punts i les seves interaccions:

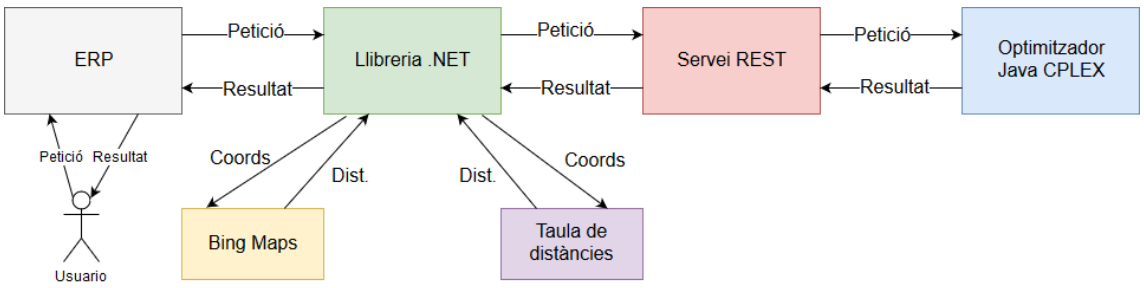


Figura 5. Esquema d'interacció entre mòduls del projecte

2.3.1 ERP

L'ERP que utilitza l'empresa és Microsoft Dynamics Axapta 2009. Aquest sistema ja està bastant antiquat i ens limita molt a l'hora d'implementar certes funcionalitats. Per exemple, la comunicació amb el servei REST s'ha de fer mitjançant una llibreria .NET 3.5.

Les dades que es volen optimitzar són extretes d'una extensa varietat de taules. Actualment aquestes taules no estan pensades per donar les dades de forma òptima. Cal dir que hi ha un projecte engegat des de fa setmanes per canviar el sistema de planificació actual de l'empresa. Aquest sistema simplificarà la gestió individual dels recursos, serà molt més flexible, i permetrà extreure els recursos i les ordres d'una forma molt senzilla i ràpida. Això implicarà una disminució important del temps de tractament de les dades a la llibreria .NET 3.5.

2.3.2 Estructura de dades

El servei REST rep les dades del problema en format JSON. Per no invertir temps en dissenyar i programar un serialitzador / des-serialitzador JSON, s'ha fet servir la llibreria Newtonsoft.Json [17] en .NET i la llibreria Jackson [18] en Java.

Per a que aquestes llibreries funcionin correctament i de forma ràpida, els diferents tipus de dades del problema han d'estar sota un mateix objecte. Les dades del problema estan formades per les següents classes. La Figura 6 mostra el diagrama de classes d'aquesta estructura de dades: (aquesta imatge es pot veure més ampliada a l'Annex 3)

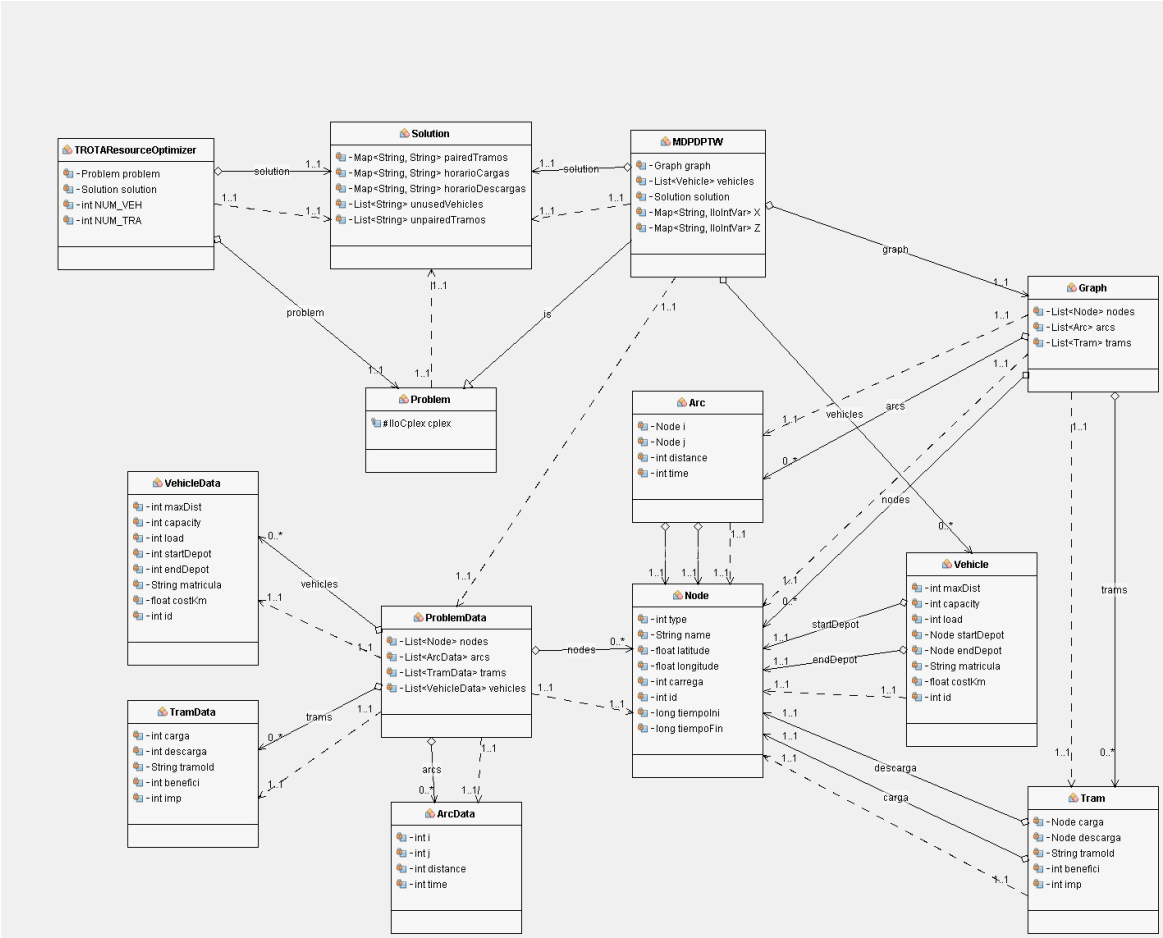


Figura 6. Diagrama de classes

- ProblemData: La classe principal que engloba les llistes de nodes, arcs, trams i vehicles. Aquesta classe és la que es passa al servei REST representada en format JSON.
- Node: Representa un node del graf (càrrega, descàrrega, inici, final). Disposa de les propietats necessàries per fer els càlculs.
- ArcData: Representa un arc dirigit del graf i les seves propietats són el node d'inici, el node destí, la distància entre l'un i l'altre i el temps que es tarda.
- VehicleData: Inclou totes les dades necessàries d'un vehicle.
- TramData: Representa una ordre d'un client. D'aquesta forma es pot associar quin arc pertany a cada ordre. També serveix per saber el nom de la ordre i paràmetres com la prioritat o el benefici d'aquesta.

Si s'observa el diagrama de classes, es pot veure que en les classes TramData, VehicleData i ArcData, els nodes implicats estan representats per tipus "int" i no per tipus "Node". Es va adoptar aquesta decisió de disseny per dos motius principals.

1. Al convertir les dades de la classe ProblemData a JSON, si es fa servir el tipus "Node" en comptes del tipus "int", es genera una enorme quantitat d'informació extra innecessària, cosa que carrega molt el tractament de les dades posteriorment. Per tant és preferible ficar només l'identificador del node.
2. Al passar les dades usant el format JSON es perden totes les referències a memòria dels objectes tipus "Node" que es troben instanciats dins dels altres objectes. Per tant es perd l'únic benefici que comportaria fer-ho d'aquesta forma.

2.3.3 Llibreria .NET

Aquesta llibreria és la intermediària entre el sistema ERP i el servei web d'optimització. Degut a l'antiguitat de l'ERP i que aquest funciona únicament al sistema operatiu Windows, la llibreria només podia ser programada usant la versió 3.5 del Framework .NET (s'ha fet servir el llenguatge C#).

Un dels aspectes més importants a decidir ha sigut la forma d'extreure les dades de l'ERP. Hi havia tres possibilitats:

1. Des de la llibreria consultar directament la base de dades que consulta l'ERP. Però va quedar descartada ja que Axapta 2009 modela d'una forma bastant particular la base de dades i qualsevol canvi a una de les taules des d'Axapta podia implicar canvis inesperats a les taules de la base de dades.
2. Des d'Axapta programar classes amb mètodes estàtics que permeten una comunicació amb el Framework .NET usant el component "Microsoft Dynamics Business Connector". Aquests mètodes farien els càlculs necessaris i retornarien les dades requerides. També es va descartar per raons d'escalabilitat i manteniment. Ja que l'ERP actual està acabant el seu cicle de vida i pot ser que a mitjà termini s'actualitzi a una nova versió. Això implicaria tornar a programar tota aquesta lògica de tractament de dades al nou ERP pro també s'hauria de canviar la llibreria. L'altra raó és que el llenguatge amb que es programa dins d'Axapta (X++) és molt lent en comparació amb el C#.
3. Vaig decidir fer servir el component "Microsoft Dynamics Business Connector" des de la llibreria per a fer les consultes a la base de dades tal i com està representada a Axapta. D'aquesta forma, si s'han de fer canvis a les taules d'Axapta, aquests serien fàcilment adaptables a la llibreria. També es pot aprofitar tot el potencial que té un llenguatge com el C# que és molt més eficient i ràpid de programar que el X++. L'únic

inconvenient que té aquest punt és que al Business Connector ja no se li dona suport des de Microsoft perquè és un component molt antic que treballa usant objectes COM. En el capítol 5 es descriu la forma en que es substituirà al Business Connector en un futur.

En la figura 7 es mostra un exemple de com funciona una consulta a la base de dades d'Axapta usant el Business Connector des de .NET 3.5. En aquest cas la consulta es fa per saber la posició d'un vehicle concret.

```
private Position     getVehiclePosition(string matricula)
{
    using (AxaptaRecord record = ax.CreateAxaptaRecord("TrafficView"))
    {
        record.ExecuteStmt("select firstonly * from %1 where %1.MatriculaTractora =='" + matricula + "'");
        string tramoId = Convert.ToString(record.get_Field("LastTramoId"));
        using (AxaptaRecord tramoRecord = ax.CreateAxaptaRecord("OVTramos"))
        {
            tramoRecord.ExecuteStmt("select firstonly * from %1 where %1.TramoId =='" + tramoId + "'");
            return getCityPosition(Convert.ToString(tramoRecord.get_Field("ToPoblacionId")));
        }
    }
}
```

Figura 7. Consulta a Axapta des de .NET

2.3.4 Càlcul de distàncies

Per a cada combinació de nodes del graf del problema es necessita saber la distància entre ells. D'aquesta forma, l'algoritme disposa del criteri principal per a escollir els camins òptims.

Per saber aquestes distàncies s'ha fet servir l'API de Bing Maps per un motiu principal. A l'empresa fa molt temps que es treballa amb software de Microsoft i ja disposem de llicències per usar aquest servei de manera il·limitada.

Consultar totes les distàncies del conjunt d'arestes A' (recordar model matemàtic), és un procés extremadament lent. Per exemple, si es vol resoldre un problema on hi ha 150 vehicles i 150 ordres a consultar, s'haurien de fer $150 * 150 + 150 + 150 * 150 = 45150$ consultes a aquest servei. El resultat d'aquest càlcul ve donat per la fórmula 5.1:

$$|SP| * |C| + |O| + |D| * |EP| \quad (5.1)$$

|SP| = Nombre de nodes del tipus "Start Depot" del graf. Sempre coincidirà amb el nombre de vehicles.

|C| = Nombre de nodes de càrrega del graf. Sempre coincidirà amb el nombre d'ordres.

|O| = Nombre d'ordres del problema a optimitzar.

|D| = Nombre de nodes de descàrrega del graf. Sempre coincidirà amb el nombre d'ordres.

|EP| = Nombre de nodes del tipus "End Depot" del graf. Sempre coincidirà amb el nombre de vehicles.

Com es pot veure, per a cada problema s'ha de fer un gran nombre de consultes. Tenint en compte que es tracten de consultes a un servei web, aquest procés encara s'alenteix més. Segons els test que he anat realitzant al llarg del projecte, puc afirmar que dins dels diferents processos del mòdul, el de calcular les distàncies és el més pesat amb diferència. Per exemple, en un problema de 150 vehicles i 150 ordres, el temps mitjà en saber totes les distàncies està al voltant de 8-9h. Tenint en compte que en el mateix problema només es

tarden entre 120 i 150 segons en generar el graf i es tarden uns 4 minuts en resoldre el problema i tornar el resultat, el temps per trobar les distàncies és extremadament costós.

Per resoldre aquest problema vaig arribar a la conclusió que tenia dues possibles solucions:

1. Paral·lelitzar les consultes a l'API de Bing Maps. Aquesta va ser la primera solució que vaig adoptar, però el fet d'haver de programar amb .NET 3.5 em va donar molts problemes ja que fins la versió 4.0 de .NET no es van implementar nativament les col·leccions amb concurrència. Totes les solucions que trobava implicaven redissenyar profundament el codi de la llibreria (cosa que no podia fer per falta de temps). Una de les solucions que vaig implementar de forma "rapida" implicava l'ús de barreres que feien perdre tot el paral·lelisme dels fils d'execució. L'altra solució que no implicava l'ús de barreres, en alguns casos, feia corrompre la col·lecció que guardava els arcs del graf.
2. La segona solució, la que he adoptat finalment, ha sigut la creació d'una base de dades de distàncies. M'he basat en el concepte "matriu de distàncies entre nodes" que és àmpliament utilitzat en aplicacions com Google Maps. Aquesta base de dades només disposa d'una taula i aquesta disposa de cinc camps. Un exemple d'aquesta taula:

		LatIni	LonIni	LatFin	LonFin	Distance
☐	Editar Copiar Borrar	35.2937	-2.9383	37.2714	-6.9495	694
☐	Editar Copiar Borrar	35.2937	-2.9383	37.2756	-6.8385	688
☐	Editar Copiar Borrar	35.2937	-2.9383	37.2829	-5.9209	597
☐	Editar Copiar Borrar	35.2937	-2.9383	37.31008	-5.96475	603

Figura 8. Exemple de la taula de distàncies.

Amb aquesta taula, el procés de càlcul de distàncies, primer busca si la clau primària (composada per les quatre coordenades), existeix a la taula. Si existeix, retorna la distància guardada anteriorment. Si no existeix, consulta l'API de Bing Maps, guarda el registre a la taula i el retorna.

La millora amb aquest sistema és excepcional, el càlcul de distàncies amb el mateix problema passa de 8-9h a 15-25 segons (depenent de la càrrega del disc dur en aquell moment).

Per altra banda, existeixen diferents inconvenients que s'han de considerar:

1. El benefici d'utilitzar aquest sistema recau en què el registre està a la base de dades i que per tant prèviament s'ha fet la mateixa petició usant l'API.
2. La utilitat d'aquest sistema va augmentant a mesura que es va fent servir, ja que cada cop hi ha mes registres guardats. Per tant, durant un temps pot anar igual de lent que sense fer-lo servir.
3. La taula pot acabar ocupant desenes de Gb en disc i pot arribar a tenir milions de registres (en dos setmanes de proves es van arribar a uns 600.000 registres).

Finalment, aquesta base de dades és MySql versió 5.6.35 i està muntada sobre un disc SSD per agilitzar les consultes. En un futur pròxim s'estudiarà utilitzar la base de dades NoSQL MongoDB muntada sobre un disc SSD i usant algun tipus de memòria cau en RAM ja que moltes de les consultes que s'intenten optimitzar al mateix dia van destinades a les mateixes zones geogràfiques.

Més endavant també s'estudiarà afegir historials de consulta i diferents tipus de comptadors d'aquestes dades per poder fer estadístiques i analitzar-les. Això pot aportar dades útils d'ús de rutes per temporades, clients, dies, etc. També servirà per a l'ús de futures heurístiques que s'apliquin abans del servei d'optimització.

2.3.5 Servei REST

Aquest servei actua com a interfície de comunicació entre la llibreria de .NET i el servei d'optimització en Java. Es buscava un sistema aprofitable en diferents entorns i que no depengués del sistema operatiu o de l'ERP. Vaig arribar a la conclusió que aquesta seria la millor forma de desacoblar l'entorn ERP amb l'entorn d'optimització. El servei web està muntat sobre un sistema operatiu CentOs 7.3 i pot ser cridat des de diferents sistemes que suportin l'ús de mètodes HTTP POST.

Per desplegar el servei web s'ha usat el Framework Spring i el gestor de dependències Maven. El gestor l'he usat per incloure l'executable ".jar" que es genera al construir el projecte de l'optimitzador. Ha sigut senzill muntar aquest servidor ja que només ha fet falta construir un controlador i un sol mètode. En la figura 9 es mostra el codi del servei REST que accepta peticions POST i retorna el resultat en format JSON.

```
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.fasterxml.jackson.databind.ObjectMapper;

import Optimizer.Problem;
import Optimizer.ProblemData;
import Optimizer.Solution;

@RestController
public class GeneralController {

    @RequestMapping(value = "/MDPDPTW/post", method = RequestMethod.POST)
    public Solution MDPDPTW(@RequestBody String jsonProblem) {
        ObjectMapper mapper = new ObjectMapper();
        Solution solution = new Solution();
        ProblemData data;

        try {
            data = mapper.readValue(jsonProblem, ProblemData.class);
            Problem mdpdptw = new Optimizer.MDPDPTW(data);
            solution = mdpdptw.solve();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return solution;
    }
}
```

Figura 9. Codi del servidor que accepta peticions POST

Cal destacar algun punt sobre aquest codi:

1. Es fa servir la llibreria Jackson per serialitzar en format JSON i per crear un objecte del tipus ProblemData.
2. La petició POST s'accepta a la URL "/MDPDPTW/post"
3. La possible excepció que llança el codi mostra un log visual a la consola del servidor que va molt bé per depurar errors, per tant de moment no s'ha canviat el auto-generat.

Un altre punt important a tenir en compte és que el servei pot acceptar peticions de forma concurrent, tot i que si són problemes gaire grans (per exemple 150 x 150), el Heap es pot quedar sense memòria en poc temps. En el capítol 3 s'explica la solució per evitar aquest problema.

2.3.6 Optimitzador

Aquest mòdul és el més important de tots. Bàsicament és el que resol el problema d'optimització i per tant entraré més en detall que en els anteriors apartats.

Primer vaig informar-me de les diferents alternatives que hi ha actualment al mercat per resoldre problemes d'optimització, tant propietàries com de codi lliure (Taules 1 i 2 a l'apartat 1.6.2). Finalment vaig decidir usar CPLEX després de veure el rendiment excepcional, la gran quantitat de documentació que existeix i per la possibilitat de disposar d'una llicència gratuïta per a estudiants.

Vaig escollir el llenguatge Java per programar el model. Pensava que a l'usar un llenguatge que ja dominava només m'hauria de centrar en l'aprenentatge de la interfície de comunicació entre CPLEX i Java. No va ser així, vaig passar-me setmanes llegint documentació i programant exemples simples per familiaritzar-me amb aquest nou tipus de programació. Em costava molt entendre perquè tots els exemples es programaven usant matrius de dades en comptes d'usar classes i objectes.

Finalment vaig acabar deduint que no era una qüestió de rendiment sinó que era una costum heretada de les persones que explicaven els exemples. Per resoldre el mètode Símplex de forma manual s'usen matrius de variables, llavors per fer que els estudiants d'aquesta matèria entenguin de forma clara els punts a seguir, ensenyen a programar els exemples de la forma més semblant a fer-ho manualment.

Des del meu punt de vista, és preferible usar estructures de dades més avançades i encapsular les variables en classes que agrupin aquestes propietats. El model matemàtic es defineix sobre un graf amb nodes i arestes i també disposa d'un conjunt de vehicles. El meu objectiu doncs, ha sigut representar aquest model de la forma més fidel possible. S'han creat diferents tipus de classes per representar els nodes, les arestes dirigides, el graf i la llista de vehicles amb totes les seves propietats.

Un dels avantatges més importants d'implementar el model matemàtic d'aquesta forma és que qualsevol canvi al model o a les restriccions és pràcticament immediat de programar. També facilita molt la comprensió del model i de les restriccions ja que totes les propietats estan tipificades i tenen un nom representatiu. En la figura 10 tenim un exemple de la implementació de les variables de decisió i de l'estructura de dades del problema:

```
private final Graph         graph = new Graph();
private List<Vehicle>       vehicles = new LinkedList<>();
private final Solution      solution = new Solution();

//variables de decisio
private final Map<String, IloIntVar> X = new HashMap<>();
private final Map<String, IloIntVar> Z = new HashMap<>();
private final Map<String, IloNumVar> S = new HashMap<>();
private final Map<String, IloNumVar> L = new HashMap<>();
```

Figura 10. Variables de decisió

Per tal de poder identificar les variables de decisió que s'han de retornar a la solució del problema, s'han representat en forma de HashMap. La clau del mapa és una combinació d'identificadors dels nodes i els vehicles implicats i el valor és la variable representada dins de la interfície de CPLEX.

En la Figura 11 es pot veure representada l'estructura de dades orientada a objectes i amb noms representatius (en comptes de fer servir matrius de variables). La imatge mostra com s'implementa la funció objectiu del model matemàtic:

```
for(Arc a : graph.getArcs()){ 1
    i = a.getI();
    j = a.getJ();
    for (Vehicle k : vehicles)2{
        sb1.delete(0, sb1.capacity());
        obj.addTerm(k.getCostKm() * a.getDistance(), X.get(sb1.
    }   5                    3          4
    if(graph.isLoad(i)){
        sb1.delete(0, sb1.capacity());              7
        t =6 graph.getTramo(i, graph.getDescargaFromCarga(i));
        obj.addTerm(t.getImp() * t.getBenefici(), Z.get(sb1.app
                            8                  9
        /*for(Vehicle k : vehicles){
            String Sname = "S_"+k.getMatricula()+"_i"+a.getI().
            obj.addTerm(1, S.get(Sname));
        }*/
    }
}
cplex.addMinimize(obj);
```

Figura 11. Funció objectiu del model

Com es pot comprovar, queda un codi on es veu clarament cada "verb" representat en el model matemàtic. Aquest model s'ha pogut simplificar respecte al model MDPDP estàndard gràcies a funcionalitats com per exemple:

- Poder representar la descàrrega equivalent a una càrrega (punt 7 de la Figura 11).
- Disposar d'una funció que retorna només els arcs dirigits d'A' (punt 1 de la Figura 11).

En la figura 12 es mostren les equivalències entre el codi font dels punts marcats a la figura 11 i els elements de la fórmula matemàtica de la funció objectiu.

$$Min\ \alpha \underset{2}{\sum_{k \in K}} \underset{1}{\sum_{(i,j) \in A'}} \underset{4}{d_{ij}} \underset{3}{L^k} X_{ijk} + \gamma \underset{5}{\sum_{i \in C}} \underset{8,9}{b_i} Z_i$$

Figura 12. Equivalències entre el model matemàtic i el codi font de la funció objectiu

Si recordem el model estàndard, les càrregues i les descàrregues es representaven de la següent manera:

- *P = {1, ..., n}* són els nodes de càrrega.
- *D = {n + 1, ..., 2n}* són els nodes de descàrrega.

Això vol dir que s'ha de disposar d'una sola matriu amb tots els nodes i s'han d'inserir seguint el model. En canvi, tal com està dissenyat actualment, els nodes poden estar distribuïts sense limitacions en el model matemàtic i les funcions com *getLoads()*, *getUnloads()* o *getDescargaFromCarga()* ens retornen els tipus de nodes desitjats de forma clara i senzilla. L'únic petit inconvenient que hi ha és que aquestes funcions no són tan eficients ni ràpides que accedint a matrius de dades. Realment no és un problema ja que la pèrdua de rendiment comparada amb altres factors és negligible.

En la Figura 13 es mostra la implementació de la restricció (4.2) del nou model matemàtic:

```
private void R1()                          throws IloException{
   StringBuilder sb1 = new StringBuilder();
    for(Node n : graph.getLoads()){ 1
        for(Vehicle k : vehicles){ 2
            IloLinearNumExpr expr = cplex.linearNumExpr();
            sb1.delete(0, sb1.capacity());
          3 expr.addTerm(1, X.get(sb1.append(k.getMatricula(
            sb1.delete(0, sb1.capacity());
          4 expr.addTerm(-1, X.get(sb1.append(k.getMatricula
            try{
              5 cplex.addEq(expr, 0, "R1");
            }
            catch(Exception ex){
                System.out.println("R1->"+ex.toString());
            }
        }
    }
}
```

Figura 13. Restricció 4.2 del nou model

Els punts 1 i 2 tornen a mostrar l'estructura de dades. En els punts 3, 4 i 5 es mostra com es fa per crear una funció restrictiva usant la interfície de CPLEX. Aquesta restricció fa que si una vehicle va des del seu punt d'inici fins a una càrrega, obligatòriament el mateix vehicle farà la descàrrega.

L'últim aspecte important a comentar és la forma en què es recupera i s'encapsula la solució del problema. Com s'ha comentat anteriorment, la variable de decisió X_{ijk} és la que acaba dient si el vehicle *k* va del node *i* al node *j*, per tant, ens diu si el vehicle *k* passa per l'arc A_{ij}.

Per saber si una variable pertany a la solució o no, no hi ha altra manera que, un cop solucionat el problema, recórrer totes les variables i anar preguntant a la interfície de CPLEX si el valor de la variable en qüestió és 1 o 0.

```
if(cplex.solve())
{
    StringBuilder sb1 = new StringBuilder();
    Node i, j;
1   for(Arc a : graph.getArcs())
    {
        i = a.getI();
        j = a.getJ();
2       for (Vehicle k : vehicles) {
            if(graph.isTramo(i, j)){
                sb1.delete(0, sb1.capacity());
                if(cplex.getValue(X.get(sb1.append(k.getMatricula()).ap
3                   solution.addPairedTramo(k.getMatricula(), graph.get
                }
            }
        }
    }

    for(Vehicle k : vehicles){
        if(!solution.getPairedTramos().containsKey(k.getMatricula())){
4           solution.addUnusedVehicle(k.getMatricula());
        }
    }
    for(Tram t : graph.getTrams()){
        if(!solution.getPairedTramos().containsValue(t.getTramoId())){
5           solution.addUnpairedTramo(t.getTramoId());
        }
    }
}
```

Figura 14. Tractament de la solució

El primer punt de la Figura 14 mostra que es recorren tots els arcs d'A'. El segon punt mostra com es recorren tots els vehicles del problema. El condicional anterior al punt 3 és on es comprova si la combinació d'un arc i un vehicle en concret és solució. En cas afirmatiu s'afegeix en un HashMap dins de l'objecte del tipus "Solution". En els punts 4 i 5 es recorren les llistes de vehicles i de trams del problema i es comprova cada ítem per si està dins del HashMap de la solució. Si no hi és, s'afegeix a la llista de vehicles o trams no emparellats.

Finalment, aquest objecte del tipus "Solution" es retorna al servei REST i aquest el serialitza en format JSON per retornar-lo a la llibreria .NET.

3 Experimentació

3.1 Validació del model

Per validar el funcionament del model i avaluar el seu comportament, es van establir dos tipus d'escenaris:

3.1.1 Escenari amb dades fabricades

Es van preparar una sèrie d'escenaris amb 4 vehicles i 3 trams per assignar. Primer es va testejar el model sense cap restricció activada i posteriorment es van anar afegint les restriccions una a una. L'ordre d'activació de les restriccions i les dades de l'escenari és el següent:

R1: Si un vehicle surt des del seu origen i va a una càrrega, posteriorment ha de fer la descàrrega corresponent.

R2: Si un vehicle va d'una càrrega a la descàrrega corresponent, posteriorment ha d'anar al destí final del vehicle (per descansar, origen d'una altra carga, etc).

R3: Si un vehicle surt des del seu origen, haurà d'acabar al seu destí.

R5: Cada carga com a molt pot ser despatxada per un vehicle.

R6: Serveix per escollir si fer sortir o no un vehicle cap a una càrrega.

R7: Per a que cada vehicle no superi la seva distància màxima que pot recórrer.

R8: Si no hi ha cap vehicle que es pugui assignar a una ordre, s'aplicarà una penalització important.

	T1	T2	T3
V1	14	65	90
V2	30	42	76
V3	62	17	34
V4	14	65	90

Taula 3. Dades de l'escenari fabricat

Fins que no es va activar la restricció R8, l'execució del programa no emparellava cap recurs. La raó d'això és que per una banda, la fórmula té la capacitat de deixar vehicles sense usar (per no obligar a crear una relació 1-1 entre trams i vehicles), per l'altra banda, l'objectiu del model és el de minimitzar els kilòmetres a recórrer, per tant, si no s'emparellen vehicles, no es recorreran kilòmetres. L'aplicació d'un gran valor de penalització (si es compara amb el nombre de kilòmetres a recórrer) per a cada tram que no es despatxa, soluciona aquest problema i llavors surt a compte "engegar" els vehicles.

Els resultats esperats (sense paràmetres de consum ni restriccions de quilometratge) eren que el vehicle 1 fos emparellat amb el tram 1, el vehicle 2 amb el tram 2 , el vehicle 3 amb el tram 3 i el vehicle 4 sense assignar.

El resultat va ser del 100% d'encert en totes les execucions que es van fer.

En la figura 15 es pot veure l'emparellament dels recursos en forma de graf (utilitzant el software "Gephi"[16]). Per generar el graf de la solució i importar-lo des de Gephi vaig implementar una exportació del resultat de les variables CPLEX en format ".gexf".

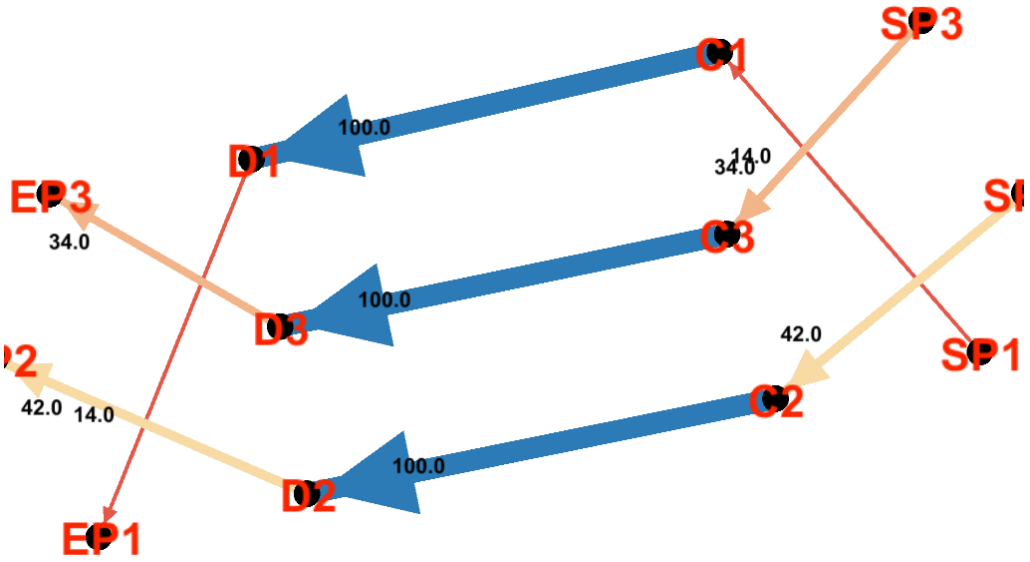


Figura 15. Solució en format de graf dirigit

Posteriorment es van afegir paràmetres a la fórmula, com el consum de combustible o el quilometratge màxim per tractora i es van tornar a fer proves. Es van anar canviant els valors dels paràmetres de forma controlada i combinada per provocar diferents situacions. Altre cop el resultat va ser l'esperat en el 100% dels casos.

Una vegada es va comprovar que el funcionament del model era correcte i no hi havia cap error espontani utilitzant dades preparades, es va procedir amb les proves amb dades reals.

3.1.2 Escenari amb dades reals

Amb dades reals "només" van fer falta tres proves per adonar-nos que el model estava incomplet i que per tant no calia seguir fent més proves fins que no s'afegissin les restriccions necessàries.

La primera prova va ser la més gran, estava composada per 243 vehicles i 199 ordres a despatxar. Bàsicament van haver quatre problemes a l'executar l'optimització d'aquest problema:

1. Es va tardar vora 5h a extreure totes les distàncies del servei de Bing Maps, cosa que ja feia bastant inútil l'optimització perquè durant aquest temps van haver diversos canvis que van fer variar els conjunts de recursos i ordres inicials. (En aquest cas hagués estat de gran ajut disposar d'una taula de distàncies bastant més completa).
2. Quan es va comprovar el resultat amb el departament de planificació, aproximadament el 85% dels recursos estaven mal emparellats perquè no estava contemplada la restricció de disponibilitat de temps de conducció dels xofers, per tant, d'haver despatxat les ordres assignades, s'haguessin passat de les hores de conducció disponibles.
3. Del 15% restant, vora un 10% no complia amb les finestres de temps establertes per arribar a la càrrega o la descàrrega, és a dir, a vegades el vehicle arribava hores abans i per tant s'havia d'esperar (desaprofitant temps de disponibilitat), i en altres ocasions el vehicle arribava tard.
4. A l'analitzar els resultats, vam poder comprovar que disposar d'uns quilòmetres màxims a recórrer per cada camió no era suficient en la nostra forma de treballar. Necessitem filar més prim i a part d'aquesta distància màxima a recórrer, també necessitem uns kilòmetres de buit màxims depenent de cada zona geogràfica,

producte i època de l'any. Sinó, hi ha casos en que els quilòmetres de buit es disparen desmesuradament i ens trobem que una càrrega que s'havia de fer al nord d'Alemanya havia de ser despatxada per un camió que sortia de buit des de Madrid.

En la segona prova que vam fer es va decidir acotar la zona geogràfica per disminuir el nombre de recursos i d'ordres a despatxar. D'aquesta manera també facilitaríem el posterior anàlisis ja que en la primera prova vam tardar més hores.

El test es va acotar a la zona d'Alemanya i va sortir un problema amb 25 vehicles i 12 ordres a despatxar. El resultat va ser que 8 dels 12 emparellaments van ser incorrectes degut a la restricció de disponibilitat dels xofers. Els 4 emparellaments restants van ser correctes i òptims pel que fa a minimització de quilòmetres i consums però des del departament de planificació em van comentar que els temps de conducció no estaven optimitzats. Això vol dir que les següents ordres que es podrien assignar a aquells vehicles haurien de ser molt a prop de les anteriors perquè la disponibilitat coincidiria amb pauses obligatòries. A planificació ho haurien emparellat d'una altra manera per tal que aquestes pauses no fossin cap problema.

Aquests comentaris van ser de gran importància. Em van fer adonar que el sistema de planificació no pot funcionar de forma òptima simplement calculant la planificació del dia següent. Per optimitzar degudament s'ha de dissenyar un sistema de planificació per fases tenint en compte fins a un màxim de 3 dies. Aquesta és la única manera de treure tot el profit a un sistema de planificació automàtic. Aquest sistema comportaria altres problemes que es comentaran a l'apartat 5.3.

Finalment, l'últim test va ser sobre el mateix problema que l'anterior però canviant paràmetres de la funció objectiu del model matemàtic. Es va anul·lar el paràmetre de combustible gastat per quilòmetre i es va modificar la penalització al no complir una ordre d'un client. D'aquesta forma l'optimització emparellaria d'una manera més semblant a com ho fa avui en dia al departament de planificació i es podrien comparar millor els resultats obtinguts. D'aquests resultats no en vam treure conclusions importants ja que eren pràcticament calcats als del segon exemple.

3.2 Anàlisis de rendiment

En aquest apartat comentaré els diferents aspectes que s'han tingut en compte per millorar el rendiment de cada apartat i algunes de les millores més significatives que s'han afegit en diferents evolucions de la implementació.

Aquest anàlisis no és exhaustiu i no mostrarà gràfics detallats de comparacions entre zones de codi en diferents situacions. Això requereix molt temps invertit i no és l'objectiu principal d'aquest projecte.

En un principi es podien optimitzar problemes de com a molt 100 vehicles i 100 ordres. El sistema tardava unes 6 hores en retornar un resultat (no hi havia la taula de distàncies implementada), això si no sortia algun error de falta de memòria abans. Avui en dia, el problema més gran que s'ha resolt ha estat de 260 vehicles i 200 ordres. Aquest tarda uns 7 minuts en resoldre's (amb la taula de distàncies prèviament emplenada) i consumeix menys memòria RAM.

El sistema on s'han executat les proves és el següent:

- Processador Intel Core i3-4150T @ 3Ghz.
- 8Gb de memòria RAM DDR3 @ 1600Mhz.
- Disc SSD Kingston SSDNow UV400 480Gb.
- Processament gràfic Intel HD Graphics 4400.

3.2.1 Tractament de dades en .NET

Totes les ordres i els vehicles que ens arriben en format de llistes d'identificadors s'han de tractar per poder crear un objecte del tipus "Problem" amb el graf del problema i totes les dades necessàries dels vehicles. Això es construeix fent centenars de consultes a la base de dades d'Axapta.

Tot i que aquesta podria semblar una operació pesada, inclús amb un problema de mida normal (150X150), no s'arriba als 10 segons en fer tot això.

3.2.2 Càlcul de distàncies

En aquest apartat sí que hi ha hagut un problema de rendiment, sobretot pel que fa al temps que es triga a fer les peticions a l'API de Bing Maps. Com s'ha comentat anteriorment, per un problema de 150x150 s'han de fer 45150 peticions a aquesta API i això podia comportar unes 8 hores. Per fer-nos una idea de com creix el nombre de crides segons la mida del problema, pel problema anterior de 260x200 s'haurien de fer 260*200+200+200*260 = 104200 peticions.

Per solucionar aquest problema es van pensar i provar varies solucions "poc" costoses en temps de programació com he comentat anteriorment però finalment s'ha optat per implementar una base de dades amb una taula de distàncies. Quan aquest sistema té totes les coordenades de les quals es necessita saber les distàncies dins de la base de dades, l'augment de rendiment és espectacular. Amb el problema de 150x150 es va passar d'unes 8h a 20 segons i en el problema de 260x200 es va passar d'unes 13h (ja hi havia algunes coordenades a la base de dades) a només 28 segons.

Cal dir que aquest sistema depèn molt del hardware ja que anirà molt més ràpid en una taula que està en un "disc dur" muntat en memòria RAM que no pas en un disc SSD o en un disc dur normal. El rendiment també pot variar considerablement depenent de la càrrega que tingui el sistema.

3.2.3 Processament de peticions al servidor

Pel que fa als problemes de rendiment d'aquest apartat no hi ha gran cosa a dir, totes les peticions que rep el servidor web són despatxades en qüestió d'uns 40-60ms. Per tant, en aquest àmbit no s'han fet modificacions per augmentar el temps de resposta.

En canvi, si que s'han tocat paràmetres del Garbage Collector de Java i del nombre màxim i mínim de memòria del Heap. Usant el Garbage Collector que hi ha per defecte el servidor es quedava penjat i es parava l'execució llençant un error. Aquest error era del tipus "OutOfMemoryError" i era degut a que en el temps d'execució del programa, més del 98% del temps era invertit en la recol·lecció de memòria i menys d'un 2% en la pròpia execució del codi.

Actualment es fa servir el Garbage Collector G1GC amb un temps màxim de pausa de 200ms (es van fer diferents proves i aquest temps va ser el més òptim). Pel que fa a la mida del Heap s'ha deixat en un mínim de 8Gb i en un màxim també de 8Gb. Com que la màquina disposa de la memòria suficient, no té cap sentit deixar una mida mínima més petita ja que

quan hagi d'ampliar-la de forma dinàmica es perdrà bastant temps. En resum, els paràmetres d'execució del servidor són els següents:

- -Xms8g
- -Xmx8g
- -XX:+UseG1GC
- -XX:MaxGCPauseMillis=200

3.2.4 Tractament de dades en Java

Dins d'aquest punt m'he anat trobant amb diversos problemes, sobretot pel que fa a la manera d'identificar les variables de decisió. Al principi, tardava vora 200 segons en introduir totes les variables dins del mapa en un problema de 130x130 (130 * (130*4) * (130*4) = 35.152.000 variables). Ràpidament em quedava sense memòria i el Garbage Collector col·lapsava l'execució.

Vaig estar bastants dies investigant com podia reduir aquest temps. Al final vaig optar per canviar totes les operacions de concatenacions de Strings per StringBuilders cosa que va fer baixar un 6-7% el temps d'execució i també el consum de memòria RAM. Més tard, fent servir el perfilador de Netbeans vaig veure que la major part del temps restant es perdia en els mètodes hash i equals que cridava el mètode put del HashMap. Vaig provar diverses alternatives que en teoria disminuïen els temps d'execució d'aquests mètodes però no van funcionar com esperava.

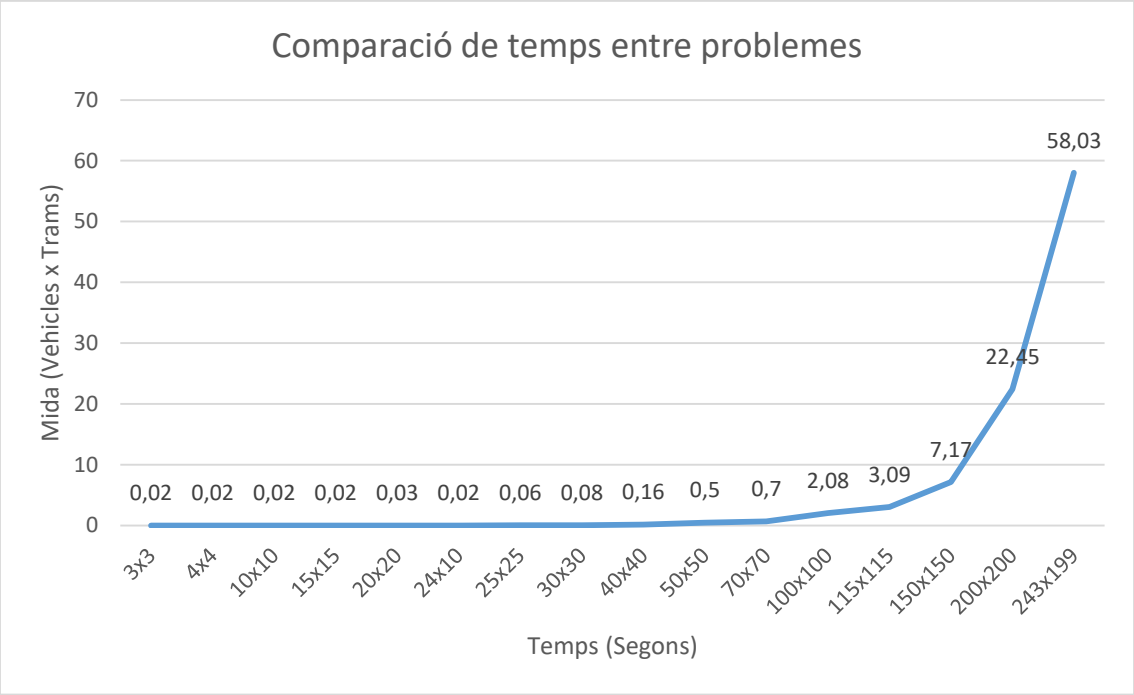
Finalment, i veient que no podia extreure més rendiment del codi, vaig decidir repassar el model matemàtic i revisar que l'estava implementant correctament. Efectivament hi havia un error important d'implementació en el model matemàtic, estava generant una variable per cada arc $a_{ij} \in A$ en comptes de generar-ne una per cada arc $a_{ij} \in A'$. Quan vaig arreglar aquest error d'implementació, el consum de memòria RAM es va reduir vora un 80% i el temps d'inserció de dades al HashMap no arribava a 10 segons. El nombre de variables a inserir va passar de 35.152.000 a 4.410.900 en un problema de 130x130.

La primera xifra surt de la fórmula |V| * |V| * m. La segona xifra surt de la fórmula (5.1) de l'apartat 2.3.5 multiplicada pel nombre de vehicles del problema.

3.2.5 Resolució del problema dins de CPLEX

En aquest àmbit no he canviat cap paràmetre de configuració. Existeixen desenes de paràmetres que es poden ajustar i que poden arribar a canviar completament el comportament de l'algoritme que soluciona el problema. Vaig estar investigant i provant durant uns dies configuracions per aquest tipus de problema en concret però no vaig notar canvis perceptibles en el temps de resolució per part de CPLEX. Per tant, vaig decidir deixar-ho tot com estava al principi.

En la Taula 5 es mostren els temps mitjans (10 intents) d'execució per a diferents mides de problemes:



Taula 4. Comparació de temps entre diferents problemes

3.2.6 Tractament de la solució

En un principi no pensava que aquest aspecte hagués de tenir un impacte important en el procés d'optimització, però ràpidament vaig adonar-me que tots els bucles que s'havien de recórrer per tractar la solució repercutirien en el temps d'execució. En un problema gran com el de 243x199 es pot arribar a tardar entre 50 i 70 segons. Abans tardava gairebé el doble perquè vaig implementar una funció que exportava el graf de la solució en un fitxer de text per a ser llegit per un software extern, d'aquesta manera era molt fàcil depurar errors en el model. Finalment vaig desactivar aquesta funció ja que no és necessària en un entorn de producció.

4 Implantació de la solució

Un procediment que no suposaria un gran impacte a la forma de treballar i que adaptaria gradualment l'automatització del sistema de planificació a l'evolució i la robustesa del model matemàtic, seria implantar aquest sistema en diferents fases. En els següents apartats es farà un breu resum de com funcionaria cada fase:

4.1 Procediment actual

Actualment, el sistema de planificació que s'està utilitzant no disposa de cap sistema d'optimització. Hi ha un departament de planificació dedicat a emparellar els recursos amb els vehicles corresponents. A hores d'ara l'ERP disposa d'un formulari on es mostren la majoria de les dades que fan falta a l'hora de planificar un viatge. Aquestes dades poden ser els períodes per fer manteniments als vehicles, els cabotatges per a cada remolc, observacions sobre els conductors, etc.

Aquest sistema comporta bastants inconvenients. És molt habitual que viatges que ja han estat planificats i assignats, s'hagin de replanificar al cap d'unes hores, i al replanificar aquests, s'han de retocar els següents que depenien d'aquests, i així fins que s'estabilitza la situació.

Aquest sistema genera una gran càrrega de treball per als planificadors que en moments puntuals poden desbordar-se per la gran quantitat d'informació que passa per les seves mans.

4.2 Procediment actual assistit

Aquesta seria la primera fase on intervindria el mòdul d'optimització de rutes. S'actuaria de la mateixa manera que en el procediment actual, però els planificadors farien una crida al sistema d'optimització i aquest els retornaria els resultats.

Un cop disposessin dels resultats, actuarien exactament de la mateixa forma que fins ara, per tant aquesta fase vindria a ser el mateix que l'anterior però descarregant de feina als planificadors, que no haurien de planificar cada viatge, sinó que només els haurien de revisar i tornar a enviar els recursos per replanificar de tant en tant.

4.3 Procediment semi-automatitzat

En aquesta fase, el model matemàtic ja seria prou robust i fiable com per poder disposar d'un sistema semi-automatitzat. Aquest hauria de ser revisat per un planificador per evitar assignacions inesperades. El procediment a seguir seria el següent:

- Hauria d'haver una sola persona encarregada del sistema de planificació.
- Aquesta persona seria l'encarregada de llançar el procés de planificació durant el dia, les vegades que fossin convenients.
- Abans de llançar el procés hauria de gestionar els recursos per afegir-los o treure'ls del grup de càlcul (per si fa falta gestionar requeriments especials que el sistema o l'algoritme encara no contemplen i impedeixen que aquell recurs estigui disponible).
- Faria la sol·licitud per l'optimització.

- Revisaria els resultats i els validaria per a que el departament de tràfic pugui veure que tenen trams pendents de gestionar (i els gestionin a les hores que siguin més convenients per si els conductors estan descansant en aquell moment, etc.)

4.4 Procediment automatitzat

Tot i que aquest seria el sistema ideal de forma teòrica, és pràcticament impossible implantar-lo per un motiu principal. No hi ha cap dia que no sorgeixi un imprevist i no s'hagi de replanificar algun viatge. Abans d'implementar aquest sistema s'hauria de fer un estudi en profunditat sobre quins són els motius que comporten aquestes replanificacions contínues i s'haurien d'aplicar accions per disminuir-les el màxim possible. En la majoria dels casos és culpa dels magatzems de càrrega o descàrrega que no s'acaba a l'hora prevista i llavors es surt més tard, s'acaben les hores de disponibilitat dels xofers, etc.

De totes formes, si s'arribés a implantar aquest procediment, tindria les següents característiques:

- Procediment completament automatitzat sense que cap persona supervisi el funcionament.
- Hi hauria un procés per lots a l'ERP que es llançaria periòdicament.
- Aquest procés recopilaria les dades necessàries de cadascun dels recursos des de diferents taules del sistema.
- Amb aquestes dades podria saber si el recurs en concret s'ha d'afegir al grup de càlcul o no pot per algun motiu.
- Es cridaria al servei d'optimització.
- El resultat obtingut es validaria automàticament per a que el departament de tràfic pogués veure els trams pendents de gestionar.

Com a apunt final a aquest capítol, cal dir que la implementació de les dos últimes fases implicaria fer canvis profunds en el sistema actual de planificació de l'empresa (tant en l'estructura de dades, la de taules com en els formularis). Aquests canvis es tenen en compte en el capítol de treball futur d'aquest projecte.

Per fer-nos una idea d'algunes de les modificacions necessàries:

- S'haurien de poder crear conjunts disjunts de recursos (es podrien afegir i treure remolcs , tractores i conductors).
- També hauria d'haver una graella des d'on es podria seleccionar el grup escollit i un botó per llançar el càlcul de la optimització.
- S'hauria de deixar processar el càlcul de forma "bloquejant" a la pantalla fins que sortís el resultat on es veuria l'emparellament entre recursos i ordres.
- El planificador hauria de poder validar els emparellaments o fer les modificacions oportunes(sempre amb recursos del mateix grup disjunt).
- S'haurien de tipificar tots i cadascun dels assumptes que es mencionen als camps oberts de "Comentaris" per tal que el sistema pogués saber en cada moment si un recurs està disponible o no.

5 Futures extensions

Aquest projecte encara té molt recorregut i aspectes que es poden millorar substancialment. Tot seguit s'explicaran els camps on es poden aplicar aquestes millores tant en el model matemàtic com en l'aprofitament de recursos computacionals del conjunt.

5.1 Nou model i noves restriccions

Per tal que el mòdul d'optimització sigui funcional respecte al model de negoci d'aquesta empresa, fan falta implementar com a mínim tres restriccions més, aquestes són:

1. Finestres de temps on s'han de despatxar tant les càrregues com les descàrregues d'una ordre. No té sentit emparellar recursos amb els mínims quilòmetres de buit si l'hora prevista per arribar a la comanda està fora del rang horari establert pel client.
2. Restriccions horàries pels temps de conducció legals dels xofers. Pot ser que un xofer acabi de fer el descans setmanal i tingui el tacògraf lliure i a un altre xofer només li quedin un parell d'hores de conducció. Per tant, a aquest últim l'algoritme l'hi hauria d'assignar un viatge molt curt o no assignar-n'hi cap.
3. Restriccions relacionades amb cabotatges. En teoria la unió europea disposa de normatives que unifiquen les lleis sobre cabotatges per tot el territori europeu. Tot i això, països com França han interpretat aquestes normatives de la forma més estricta possible i disposen d'unes lleis molt restrictives de cabotatge dins del seu territori. S'hauria d'implementar una restricció que tingui en compte aquest punt per evitar multes o inclús la retirada de la llicència de transport.

També farien falta altres tipus de restriccions que no són essencials pel funcionament de la planificació però que encara refinarien més el sistema d'optimització. Per exemple:

1. Els vehicles que no disposen d'endoll i porten una càrrega amb congelats, no podran anar per rutes que intervingui un vaixell transbordador o "Ferry".
2. S'haurà d'implementar un màxim de quilòmetres de buit variable depenent del període anual, del producte transportat i de la zona on s'ha de moure el vehicle.

5.2 Implementació d'una heurística o d'una metaheurística

Tal com he anat comentant al llarg del document, optimitzar d'una sola vegada un problema real del món del transport és inviable. Per poder fer-ho es necessitaria disposar d'un sistema hardware amb una gran capacitat de càlcul i memòria que suposaria una inversió injustificada.

Existeix una forma per simplificar aquest problema en diferents subconjunts més senzills. La solució seria implementar una heurística (o una metaheurística) que en un temps reduït i segons un criteri especificat (per exemple la distància o la latitud i la longitud) faria agrupacions de vehicles i ordres. Un cop fetes aquestes agrupacions, es llançarien de forma paral·lela tantes crides al mòdul d'optimització com grups s'haguessin creat.

A la tesis de Pairoj Chaichiratikul (2013)[1] s'estudia precisament l'aplicació d'una meta heurística hibrida per solucionar aquest problema.

Avui en dia s'està estudiant de forma molt activa l'ús d'algoritmes memètics, heurístiques i meta heurístiques en aquest camp i existeix una quantitat immensa de documentació acadèmica al respecte.

5.3 Planificació usant programació estocàstica multi-etapa

Per a aprofitar al màxim el sistema d'optimització i planificació automàtic, s'ha de tenir en compte que no es planifica només pensant en el següent dia. Com s'ha esmentat anteriorment, el departament de planificació, a part de minimitzar els kilòmetres de buit, també intenta optimitzar els temps de conducció dels xofers per a que no els coincideixin pauses obligatòries amb les següents ordres.

L'única manera d'optimitzar les hores de conducció és tenint en compte totes les ordres possibles (inclús previsions cícliques) que hi haurà durant els pròxims dies.

La forma més senzilla d'implementar això és indicant els temps d'inici i final previstos per a cada ordre en forma de "ticks" passats des d'un instant de temps concret. Així doncs es podria resoldre el problema d'una sola vegada però seria un problema "gegant" i seria molt difícil escollir criteris encertats per a les agrupacions de les heurístiques.

L'altra forma, més complexa però més eficient, seria intentar optimitzar el problema usant programació estocàstica multi-etapa. Aquest procés implicaria provar un nombre finit i predeterminat de diferents combinacions d'estats en varies fases.

Per exemple, es pot fixar que per cada fase s'intentaran 10 combinacions possibles de resultats òptims, cada combinació representaria l'inici del nou problema d'optimització de la següent fase, que intentaria optimitzar les ordres del següent dia, i així fins el nombre de fases (dies) que hi hagi al problema. Llavors s'agafaria la combinació de fases més òptima.

5.4 Canvi d'estructura de dades dins l'ERP

Des del meu punt de vista, crec que és molt necessari desvincular les limitacions de l'extracció de dades de l'ERP del sistema de càlcul de distàncies i optimització.

Com he anat comentant al llarg del document, Axapta 2009 ja està antiquat i està molt limitat en varis aspectes. Un d'aquests es que ens obliga a usar la tecnologia .NET 3.5 per comunicar-nos amb el servei d'optimització.

El primer pas per arreglar això és canviar el flux de treball de la planificació de recursos dins l'ERP. En aquest sentit ja fa temps que s'hi està treballant dins del departament d'informàtica. També s'està redissenyant tota l'estructura de dades i de taules per a atomitzar els recursos i poder fer vinculacions molt més dinàmiques. La nova estructura de taules s'ha pensat per a que sigui molt senzill i ràpid extreure dades des de processos tant interns com externs.

Un cop estigui aquest sistema implementat, per desvincular les limitacions anomenades anteriorment s'hauria de dissenyar un altre procés. Aquest procés implicaria:

1. Crear una base de dades nova, independent de l'ERP, per exemple usant MySQL sobre un sistema operatiu Linux.
2. Crear les taules necessàries de recursos i ordres disponibles per optimitzar, per exemple la taula de tractores, la de remolcs, la de conductors i la de trams disponibles.
3. Cada certs minuts / hores, des de l'ERP passarà un procés que escriurà a la BBDD externa els recursos que estiguin disponibles i les ordes que es vulguin emparellar als recursos.

4. Al llançar el procés d'optimització des de l'ERP es cridarà una llibreria programada en C++ que simplement tindrà una funció bloquejant externa la qual agafarà les dades de la nova BBDD, les empaquetarà en format JSON i cridarà al servei REST.
5. Quan la llibreria rebi les dades, escriurà els resultats a la BBDD externa i un cop acabada la escriptura desbloquejarà la crida a la funció des de l'ERP.
6. Des d'Axapta es llegirà la taula dels resultats de la BBDD externa i es tractarà de la forma més convenient.

Amb aquests nous processos haurem aconseguit simplificar molt la forma d'extreure dades, fer-la molt més adaptable a canvis i sobretot desvincular les limitacions tecnològiques de l'ERP.

5.5 Canvi de llenguatge del mòdul d'optimització

Ara mateix el mòdul d'optimització està programat en Java. Tot i que en els últims anys aquest llenguatge i el seu entorn (compilador, garbage collector, etc) han millorat de forma excepcional el rendiment i l'ús de memòria, encara està lluny de llenguatges de més baix nivell com el C o el C++.

Es va escollir Java per la simplicitat i l'alt grau de productivitat de programació tenint en compte el temps que es disposava per fer el projecte.

Una de les millores més importants pel que fa al rendiment i a l'ús de memòria seria passar el model matemàtic a C++. De fet, durant una setmana vaig intentar passar el mòdul d'optimització de Java a C++ i vaig haver de parar el desenvolupament perquè vaig veure que la modelització del problema s'havia de programar de forma completament diferent. Ja no per les adaptacions del llenguatge sinó perquè la interfície de CPLEX de Java i C++ són bastant diferents. Vaig estar documentant-me i practicant amb exemples, però era impossible convertir-ho en un període de temps raonable.

El millor que es podria fer és el dia que es decideixi remodelar el problema i afegir-hi les noves restriccions, és programar-ho tot en C++ directament.

5.6 Prescindir de CPLEX

En una etapa avançada del projecte, un cop es disposi d'un model robust i completament útil per a l'empresa, es podria començar a pensar en la substitució de la suite de CPLEX. Bàsicament hi ha dos motius principals pels quals fer aquest pas:

1. No haver de pagar la llicència de més de 15.000€ anuals per l'ús d'aquesta suite. Per aprofitar encara més la disminució de costos.
2. Dissenyar un algoritme més adequat per a ser paral·lelitzat en hardware de processament de gràfics (GPU), com a exemple.

Respecte al segon punt he estat llegint bastanta documentació (classificada a la bibliografia). Durant els últims anys, sobretot des de que NVIDIA va treure al mercat el sistema CUDA, han aparegut nombroses tesis investigant la paral·lelització de l'algoritme Símplex. Algunes d'elles han set exitoses però moltes no han obtingut els resultats esperats. Això és degut a la gran quantitat de missatges que s'han de passar entre el processador i les targetes gràfiques. Els casos d'èxit han estat de tesis amb un grau molt elevat de complexitat, de coneixement del hardware i de les llibreries CUDA.

Des de l'any 2010 NVIDIA disposa d'una tecnologia anomenada GPUDirect que permet reduir de forma dramàtica la càrrega a la CPU, la latència i la quantitat de memòria copiada innecessàriament en un clúster de GPU's. L'any 2013 van millorar aquest sistema amb la tecnologia GPU Direct RDMA. Aquesta nova tecnologia permet als dispositius de tercers que fan servir PCI Express poder accedir directament a la GPU i a la seva memòria sense dependre de la CPU.

Lògicament, el disseny i la programació d'aquest nou algoritme hauria de tenir en compte aquests nous avanços tecnològics.

Per altra banda, també hi ha la possibilitat d'explotar la paral·lelització de les dades i processos utilitzant infraestructures clúster o cloud.

5.7 Canviar el sistema de la base de dades de distàncies

Existeixen dues possibles millores en aquest punt. Per una banda, la base de dades només disposa d'una taula. Aquesta fa servir un sistema de clau-valor, la clau primària està formada per quatre camps i un valor que és la distància. La millor alternativa seria canviar a una base de dades NoSQL com MongoDB que està més preparada per aquest tipus de taules.

Pel que fa al hardware crec que hi ha dues possibles millores:

1. Muntar la BBDD en un disc SSD o en un sistema RAID de SSD per millorar la tassa de lectura / escriptura de registres. Cal recordar que el càlcul de distàncies és un dels apartats més costosos en temps.
2. Muntar la BBDD en un disc emulat en memòria RAM que faci copies periòdiques en un disc dur per si hi ha algun tipus de problema. La millora de rendiment seria considerablement més alta respecte al punt anterior però a mesura que la base de dades s'anés expandint s'hauria d'expandir la quantitat de memòria RAM del sistema.

6 Conclusions

6.1 Conclusions sobre els objectius

Pel que fa als objectius del projecte referents al període de duració del Treball de Fi de Màster puc dir que estic plenament satisfet. He aconseguit crear el mòdul d'optimització que era la tasca principal. Per fer-ho, he hagut d'estudiar profundament i partint des de zero com funciona tot el món de l'optimització, l'estat de l'art, algoritmes, softwares, etc.

A més a més d'aquesta part teòrica, he hagut de dissenyar i implementar la millor forma possible per dur-ho a terme (tenint en compte les restriccions tecnològiques) i he acabat integrant entre sí una gran varietat de tecnologies. Per tant, crec que la part acadèmica del projecte ha sigut satisfactòria.

Per altra banda, no he quedat massa satisfet amb el nivell d'utilitat al qual s'ha arribat respecte al model matemàtic. A mesura que avançava amb el projecte m'anava animant i pensava que a l'acabar aquest treball, el model estaria implementat de forma que ja podria ser d'ajuda al departament de planificació. Com s'ha comentat anteriorment, això no ha sigut possible, encara falten afegir tres restriccions bàsiques al model abans que puguem començar a utilitzar-lo. Per intentar resoldre això, durant uns dies vaig estar implementant la restricció de les finestres de temps però vaig arribar a la conclusió de que requeria un estudi molt més profund que quedava fora dels objectius d'aquest treball.

6.2 Conclusions personals

Sempre he sigut una persona que m'ha agradat "optimitzar" mentalment petits problemes del dia a dia i ja feia bastants anys que m'havia interessat per saber com funcionava tot aquest món en l'àmbit acadèmic i empresarial.

Per diferents motius com poden ser la falta d'un objectiu concret, la gran inversió de temps que es necessita per treballar amb un tema d'aquesta magnitud o perquè realment no sabia per on començar, no havia profunditzat mai en aquest àmbit i era una assignatura que encara tenia pendent.

Estic molt satisfet d'haver començat aquest projecte i sincerament crec que són de les millors hores que invertit en la meva vida per les següents raons:

- He resolt una necessitat personal sobre la qual tenia moltíssima curiositat des de fa anys.
- No m'havia sentit mai tant motivat ni havia disposat mai de tanta concentració i energia com la que m'ha aportat el realitzar aquest projecte. Hi ha hagut dies que m'he passat tota la jornada laboral amb el projecte, a l'arribar a casa he seguit amb ell fins l'hora d'anar a dormir i estava desitjant despertar-me per tornar-m'hi a posar.
- Crec que l'experiència d'aquest projecte m'ha obert moltes portes en sectors tant acadèmics com industrials on facin falta persones qualificades per avançar estudis o implementacions d'aquest àmbit.
- Durant el treball, m'han sorgit diverses idees de projectes sobre les quals seguiré treballant en el temps personal.
- M'ha obert les portes a poder fer un doctorat industrial implementant el treball futur esmentat al capítol 5 aprofitant les dades reals d'una empresa del sector.

En resum, crec que he complert amb els objectius del treball, tant acadèmics com personals però sóc conscient que encara queda molta feina a fer i molta matèria per seguir estudiant.

7 Fonts i bibliografia consultada

[1] Chaichiratikul, P. (2017). *Hybrid metaheuristics for solving multi-depot pickup and delivery problems*. [online] Spiral.imperial.ac.uk. Available at: https://spiral.imperial.ac.uk/handle/10044/1/30647 [Accessed 10 Nov. 2017].

[2] Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J. and Mahajan, A. (2013). Mixed-integer nonlinear optimization. *Acta Numerica*, 22, pp.1-131.

[3] Bernabe.dorronsoro.es. (2016). *VRP Web*. [online] Available at: http://www.bernabe.dorronsoro.es/vrp/index.html?/Problem_Descriptions/VRPDesc.html [Accessed 10 Nov. 2016].

[4] Es.wikipedia.org. (2016). *Optimización (matemática)*. [online] Available at: https://es.wikipedia.org/wiki/Optimizaci%C3%B3n_(matem%C3%A1tica) [Accessed 3 Nov. 2016].

[5] Genetic Algorithm for Multi-Criteria Optimization of Multi-Depots Pick-up and Delivery Problems with Time Windows and Multi-Vehicles. (2015). *Acta Polytechnica Hungarica*, 12(08).

[6] Kachitvichyanukul, V., Sethanan, K. and Golinska-Dawson, P. (n.d.). *Toward sustainable operations of supply chain and logistics systems*. 1st ed.

[7] LAU, H. and LIANG, Z. (2002). PICKUP AND DELIVERY WITH TIME WINDOWS: ALGORITHMS AND TEST CASE GENERATION. *International Journal on Artificial Intelligence Tools*, 11(03), pp.455-472.

[8] Lau, H., Sim, M. and Teo, K. (2003). Vehicle routing problem with time windows and a limited number of vehicles. *European Journal of Operational Research*, 148(3), pp.559-569.

[9] Pinedo, M. (2005). *Planning and scheduling in manufacturing and services*. 1st ed. New York, NY: Springer.

[10] YouTube. (2016). *CPLEX & Java 4. Travelling salesman problem*. [online] Available at: https://www.youtube.com/watch?v=QzOLL2tUXKE [Accessed 4 Nov. 2016].

[11] YouTube. (2016). *CPLEX Seminar - Solving the VRPTW in Java with column generation*. [online] Available at: https://www.youtube.com/watch?v=KQxQiKbGB5E [Accessed 5 Nov. 2016].

[12] Nvidia. (2017). *GpuDirect.* [online] Available at: https://developer.nvidia.com/gpudirect 10/01/2017 [Accessed 10 Feb. 2017].

[13] Simplex. (2017). *Método Simplex.* [online] Available at: https://www.ingenieriaindustrialonline.com/herramientas-para-el-ingeniero-industrial/investigaci%C3%B3n-de-operaciones/m%C3%A9todo-simplex/ [Accessed 12 Feb. 2017].

[14] Phpsimplex.com. (2017). *Teoría del Método Simplex*. [online] Available at: http://www.phpsimplex.com/teoria_metodo_simplex.htm [Accessed 2 Mar. 2017].

[15] Maven.apache.org. (2017). *Maven – Introduction*. [online] Available at: https://maven.apache.org/what-is-maven.html [Accessed 19 Feb. 2017].

[16] Gephi.org. (2017). *Gephi - The Open Graph Viz Platform*. [online] Available at: https://gephi.org/ [Accessed 3 Jan. 2017].

[17] Newtonsoft.com. (2017). *Json.NET - Newtonsoft*. [online] Available at: http://www.newtonsoft.com/json [Accessed 4 Jan. 2017].

[18] GitHub. (2017). *FasterXML/jackson*. [online] Available at: https://github.com/FasterXML/jackson [Accessed 5 Jan. 2017].

[19] SCIP. (2017). *Scip*. [online] Available at: http://scip.zib.de [Accessed 15 Apr. 2017].

[20] Gurobi. (2017). *Gurobi*. [online] Available at: http://www.gurobi.com/products/features-benefits [Accessed 15 Apr. 2017].

[21] Wolfram. (2017). *Mathematica*. [online] Available at: http://www.wolfram.com/mathematica/ [Accessed 15 Apr. 2017].

8 Glossari

CPU: Central Processing Unit. Unitat de Processament Central

GPU: Graphics Processor Unit. Unitat de Processament de Gràfics

RAID: Redundant Array of Inexpenseive Disks. Array Redundant Barat de Discs.

RAM: Random Access Memory. Memòria d'Accés Aleator.

SSD: Solid State Disk. Disc d'Estat Sòlid.

ERP: Enterprise Resource Planning. Planificador de Recursos Empresarials.

SQL: Structured Query Language. Llenguatge de Consulta Estructurat.

Annex 1 – Problema en format JSON

El següent exemple mostra la representació en format JSON d'un problema on hi ha tres vehicles i tres ordres a emparellar. Per reduir el nombre de pàgines de l'exemple s'ha estructurat en dues columnes.


```
{
"nodes": [{
"type": 0,
"name": "spIF43TRO",
"latitude": 51.0215,
"longitude": 3.20927,
"carrega": 0,
"id": 0,
"tiempoIni": 0,
"tiempoFin": 0
}, {
"type": 3,
"name": "epIF43TRO",
"latitude": 51.0215,
"longitude": 3.20927,
"carrega": 0,
"id": 1,
"tiempoIni": 0,
"tiempoFin": 0
}, {
"type": 0,
"name": "spIF44TRO",
"latitude": 50.44028,
"longitude": 8.79617,
"carrega": 0,
"id": 2,
"tiempoIni": 0,
"tiempoFin": 0
}, {
"type": 3,
"name": "epIF44TRO",
"latitude": 50.44028,
"longitude": 8.79617,
"carrega": 0,
"id": 3,
"tiempoIni": 0,
"tiempoFin": 0
}, {
"type": 0,
"name": "spIF15TRO",
"latitude": 38.28145,
"longitude": -1.45682,
"carrega": 0,
"id": 4,
"tiempoIni": 0,
"tiempoFin": 0
}, {
"type": 3,
"name": "epIF15TRO",
"latitude": 38.28145,
"longitude": -1.45682,
"carrega": 0,
"id": 5,
"tiempoIni": 0,
"tiempoFin": 0
}, {
"type": 1,
"name": "PBL-005963",
"latitude": 52.2946,
"longitude": 8.8976,
"carrega": 0,
"id": 6,
"tiempoIni": 7362294,
"tiempoFin": 7362414
}, {
"type": 2,
"name": "PBL-005528",
"latitude": 49.0726,
"longitude": 3.4243,
"carrega": 0,
"id": 7,
"tiempoIni": 7367810,
"tiempoFin": 7367930
}, {
"type": 1,
"name": "PBL-019314",
"latitude": 49.2717,
"longitude": -1.0149,
"carrega": 0,
"id": 8,
"tiempoIni": 7888854,
"tiempoFin": 7888974
}, {
"type": 2,
"name": "PBL-134396",
"latitude": 43.771,
"longitude": 1.3578,
"carrega": 0,
"id": 9,
"tiempoIni": 7892465,
"tiempoFin": 7892585
}, {
"type": 1,
"name": "PBL-142529",
"latitude": 50.8083649,
"longitude": -1.09154272,
"carrega": 0,
"id": 10,
"tiempoIni": 8417559,
"tiempoFin": 8417679
}, {
"type": 2,
"name": "PBL-131745",
"latitude": 48.7461,
"longitude": 2.3527,
"carrega": 0,
"id": 11,
"tiempoIni": 8419500,
"tiempoFin": 8419620
}],
"arcs": [{
"i": 6,
"j": 7,
"distance": 649,
"time": 360
}, {
"i": 8,
"j": 9,
"distance": 813,
"time": 478
}, {
"i": 10,
"j": 11,
"distance": 574,
"time": 398
}, {
"i": 0,
"j": 6,
```

```
"distance": 485,
"time": 264
}, {
"i": 0,
"j": 8,
"distance": 500,
"time": 282
}, {
"i": 0,
"j": 10,
"distance": 395,
"time": 291
}, {
"i": 2,
"j": 6,
"distance": 325,
"time": 164
}, {
"i": 2,
"j": 8,
"distance": 903,
"time": 476
}, {
"i": 2,
"j": 10,
"distance": 860,
"time": 522
}, {
"i": 4,
"j": 6,
"distance": 2118,
"time": 1166
}, {
"i": 4,
"j": 8,
"distance": 1655,
"time": 907
}, {
"i": 4,
"j": 10,
"distance": 2127,
"time": 1236
}, {
"i": 7,
"j": 1,
"distance": 303,
"time": 172
}, {
"i": 7,
"j": 3,
"distance": 534,
"time": 277
}, {
"i": 7,
"j": 5,
"distance": 1656,
"time": 918
}, {
"i": 9,
"j": 1,
"distance": 926,
"time": 507
}, {
"i": 9,
"j": 3,
"distance": 1199,
"time": 642
}, {
"i": 9,
"j": 5,
"distance": 848,
"time": 545
}, {
"i": 11,
"j": 1,
"distance": 286,
"time": 167
}, {
"i": 11,
"j": 3,
"distance": 624,
"time": 326
}, {
"i": 11,
"j": 5,
"distance": 1558,
"time": 863
}],
"trams": [{
"carga": 6,
"descarga": 7,
"tramoId": "TOV-332199",
"benefici": 10000,
"imp": 10
}, {
"carga": 8,
"descarga": 9,
"tramoId": "TOV-432145",
"benefici": 10000,
"imp": 10
}, {
"carga": 10,
"descarga": 11,
"tramoId": "TOV-553523",
"benefici": 10000,
"imp": 10
}],
"vehicles": [{
"maxDist": 7323,
"capacity": 33,
"load": 0,
"startDepot": 0,
"endDepot": 1,
"matricula": "IF43TRO",
"costKm": 0.1253131,
"id": 0
}, {
"maxDist": 5399,
"capacity": 33,
"load": 0,
"startDepot": 2,
"endDepot": 3,
"matricula": "IF44TRO",
"costKm": 0.119393744,
"id": 1
}, {
"maxDist": 5399,
"capacity": 33,
"load": 0,
"startDepot": 4,
"endDepot": 5,
"matricula": "IF15TRO",
"costKm": 0.111782514,
"id": 2
}]
}
```

Annex 2 – Exemple de visualització de la solució

En aquest annex es mostra un exemple de com es veu per pantalla part de la solució d'un problema (mostrar-la tota era innecessàriament llarg). La primera part mostra el que fa CPLEX per trobar la solució i en la segona es mostra la solució en sí.

```
Found incumbent of value 1.9900000e+007 after 5.27 sec. (864.76 ticks)
Tried aggregator 2 times.
MIP Presolve eliminated 48800 rows and 23404987 columns.
MIP Presolve modified 106 coefficients.
Aggregator did 96714 substitutions.
Reduced MIP has 441 rows, 48357 columns, and 96515 nonzeros.
Reduced MIP has 48357 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 49.02 sec. (9940.16 ticks)
Tried aggregator 1 time.
Reduced MIP has 441 rows, 48357 columns, and 96515 nonzeros.
Reduced MIP has 48357 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.34 sec. (128.02 ticks)
Probing time = 0.11 sec. (16.81 ticks)
Clique table members: 441.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 4 threads.
Root relaxation solution time = 0.31 sec. (191.54 ticks)

      Nodes                                         Cuts/
 Node Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*    0+    0                    1.99000e+007        0.0000              100.00%
*    0+    0                    68996.7889          0.0000              100.00%
*    0+    0                    68798.3962          0.0000              100.00%
*    0     0      integral  0    62482.7020      62482.7020     1656    0.00%
Elapsed time         = 58.61 sec. (12460.70 ticks, tree = 0.00 MB, solutions = 4)

Root node processing (before b&c):
Real time            = 65.16 sec. (13629.13 ticks)
Parallel b&c, 4 threads:
Real time            = 0.00 sec. (0.00 ticks)
Sync time (average)  = 0.00 sec.
Wait time (average)  = 0.00 sec.
                       ------------
Total (root+branch&cut) =   65.16 sec. (13629.13 ticks)
----------------------------------------------------
TOTAL
[62482.702007614076]

EMPARELLAMENTS
IF96TRO  ->          NCC-000490/17
IF73TRO  ->          ICC-002879/17
IF50TRO  ->          ECC-002181/17
8271JFZ  ->          ICC-003181/17
IF21TRO  ->          ECC-002313/17
1918JHZ  ->          ICC-003090/17
IF15TRO  ->          ECC-002102/17
IF75SIT  ->          ICC-003088/17
IF98SIT  ->          ICC-003242/17
9567JHX  ->          ECC-000943/17
5417JDM  ->          ICC-002908/17
IF35SIT  ->          NCC-000487/17
IF67TRO  ->          ICC-003189/17
2993JGF  ->          ECC-002122/17
IF58SIT  ->          ICC-003505/17
3126JHW  ->          ECC-002180/17
9320JHX  ->          NCC-000565/17
5731JHG  ->          NCC-000486/17
IF26TRO  ->          ICC-003084/17
```

9538JHX -> ECC-002353/17
IF78TRO -> ICC-002837/17
0588JHF -> ICC-003500/17
IF91TRO -> ECC-002259/17
IF87SIT -> ICC-003386/17
IF69SIT -> ECC-002175/17
5823JHC -> ECC-002168/17
5400JDM -> NCC-000508/17
7002JGD -> NCC-000560/17
IF17SIT -> ECC-001853/17
6990JGD -> ICC-003526/17
5698JHG -> ICC-002853/17
3125JHW -> NCC-000551/17
IF79TRO -> ECC-000938/17
IF47SIT -> ICC-003380/17
(...)
IF84TRO -> ECC-002197/17
IF08TRO -> ICC-003076/17
IF16FRA -> ICC-003117/17
6612JDS -> ICC-002571/17
6003JHC -> ICC-003106/17
4682JHC -> NCC-000522/17
IF45TRO -> ECC-002138/17
IF20TRO -> ECC-002139/17
IF64SIT -> ECC-002101/17
6635JDS -> NPA-000004/17
9391JHX -> ECC-002452/17
IF81SIT -> ICC-003203/17
IF62TRO -> ECC-002482/17
IF14TRO -> ECC-002257/17
9860JTW -> ECC-002264/17

VEHICLES SENSE USAR
6158JHC
5406JDM
5244JTZ
5217JTZ
6616JDS
6655JTW
5432JDM
(...)
9855JTW
5354JDM
6637JDS
1976JHZ
6632JDS
IF94SIT
9852JTW
5378JDM
9258JHX

TRAMS SENSE ASSIGNAR

Annex 3 - Diagrama de classes